HPI Future SOC Lab - Proceedings 2019

Christoph Meinel, Andreas Polze, Karsten Beins,
Rolf Strotmann, Ulrich Seibold, Kurt Rödszus,
Jürgen Müller (Eds.)

Technische Berichte Nr. 158

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 158

Christoph Meinel | Andreas Polze | Rolf Strotmann | Ulrich Seibold | Kurt Rödszus |
Jürgen Müller (Eds.)

HPI Future SOC Lab

Proceedings 2019

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2024
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Druck: docupoint GmbH Magdeburg

ISBN 978-3-86956-564-4

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
https://doi.org/10.25932/publishup-59791
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-597915

Preface

The *HPI Future SOC Lab* is a cooperation of the Hasso Plattner Institute (HPI) and industry partners. Its mission is to enable and promote exchange and interaction between the research community and the industry partners.

The HPI Future SOC Lab provides researchers with free of charge access to a complete infrastructure of state of the art hard and software. This infrastructure includes components, which might be too expensive for an ordinary research environment, such as servers with up to 64 cores and 2 TB main memory. The offerings address researchers particularly from but not limited to the areas of computer science and business information systems. Main areas of research include cloud computing, parallelization, and In-Memory technologies.

This technical report presents results of research projects executed in 2019. Selected projects have presented their results on April 9th and November 12th 2019 at the Future SOC Lab Day events.

Contents

Scalable Global Membarriers for Multicore Systems

Robert Kuban[1], Randolf Rotta[1], and Jörg Nolte[1]

Distributed Systems & Operating Systems
Brandenburgische Technische Universität Cottbus-Senftenberg
{firstname.lastname}@b-tu.de

Multicasts are a crucial component of consistency mechanisms. For example, some memory reclamation protocols use multicasts to observe the memory reservations of all cores without frequent costly fence instructions. Highly dynamic receiver groups incur a high overhead for maintaining multicast trees. Thus, sequential propagation is used in practice, which results in linear scaling latency. This project evaluates a mechanism that archives logarithmic scaling without continuously maintaining multicast trees. The impact on latency and throughput is evaluated with user-space Read-Copy-Update benchmarks by extending the membarrier system call of the Linux kernel. On sufficiently large systems, these strategies outperform the conventional sequential propagation with reasonable impact on the throughput.

1 Introduction

The membarrier multicast was introduced in the context of the user-space Read-Copy-Update library.[1] It interrupts all cores that are currently used by the target process in order to perform a local full memory barrier [6] such as the `mfence` instruction on x86 processors. The multicast is synchronous by waiting for an acknowledgement from each affected core.

The main use of the membarrier multicast is to replace frequent but slow full memory barriers by much faster compiler barriers. Just when a core actually needs to observe the other cores' state, it issues the membarrier multicast in order to perform the missing memory barrier on each core. This approach is, for example, applied in uRCU to speed up readers because the read lock and unlock primitives can avoid the costly full memory barrier. Just the infrequent update operations actually observe these locks through applying the membarrier multicast.

The membarrier system call was proposed by Mathieu Desnoyers[2] and it was included in the Linux kernel with version 4.3. Memory reclamation schemes that publish memory reservations, for example with hazard pointers [3] or intervall-based memory reclamation [9], can also benefit from the membarrier multicast. Finally, the TLB shootdown, which invalidates address translation caches of affected

[1] http://liburcu.org/ (last accessed 2020-04-01).
[2] https://lkml.org/lkml/2010/1/6/500 (last accessed 2020-04-01).

cores, uses a similar mechanism and, thus, can benefit from faster multicasts. In fact, Dice, Herlihy and Kogan [3] used the TLB shootdown mechanisms as an improvised memory barrier.

Although the membarrier multicast helps to increase the throughput on the fast path, the latency impact of the multicast itself on the slow path should not be neglected: The time needed to identify all relevant cores, to interrupt them for their local memory barrier, and to collect their acknowledgements increases with the number of cores. For instance, on a 72-core system, a multicast across all cores took around 20μs. For comparison, MPI's latency for small messages is in the range of 1-2μs via Infiniband nowadays.[3] In theory, multicasts require just logarithmically growing latency by using trees [1, 4, 5, 7]. However, the highly dynamic receiver groups incur a high overhead for maintaining such trees and, thus, sequential propagation with just linear scaling is used in practice. If, for example, a sequence of RCU updates requires ordering, at least one membarrier multicast is needed after each update. Thus, the huge multicast completion latency easily dominates the latency of the overall update operation.

This paper reports the evaluation of algorithms that reduce the latency of the membarrier multicast. The recursive construction of multicast trees of Bruck et al. [2] is adapted to the Linux kernel with dynamic groups of cores and compared against the conventional sequential propagation. The results show that the on-the-fly creation of multicast trees can indeed reduce the multicast latency and enable logarithmic scaling. However, the throughput scalability for concurrent multicasts is slightly worse than with the sequential propagation.

2 Shared Memory Multicast Implementations

This section outlines two different approaches to implementing multicasts in shared memory. The first is the conventional approach of the Linux kernel, which sequentially propagates the messages and acknowledgements. Subsequently, an approach that propagates messages along an on-the-fly binomial tree topology is described.

2.1 Conventional Sequential Propagation

The Linux implementation of the membarrier system call sequentially scans the mask of online cores for cores that have to participate in the full memory barrier. Then, it uses `call_function_many` to enqueue a small task in the task queue of each receiver and to send an inter-processor interrupt (IPI). The interrupt handler executes the enqueued tasks. The membarrier task just performs the local full memory barrier and acknowledges its completion by setting a flag in the task.

[3] See, for example http://mvapich.cse.ohio-state.edu/performance/pt_to_pt/.

Consequently, the execution of the tasks is carried out in parallel, but sending the tasks and collecting the acknowledgements is sequential. Thus, the membarrier multicast latency should scale linearly with the number of cores.

2.2 On-The-Fly Multicast Tree

Our implementation of the membarrier system call replaces the sequential multicast mechanisms against a tree-based multicast for dynamic receiver groups. Each process requires an individual multicast tree, which spans all cores that currently execute a thread of the process. Updating these trees whenever the scheduler switches between processes would incur a huge overhead. Instead, the multicast tree is constructed on the fly and in parallel just when needed.

As it turns out, the recursive construction of binomial multicast trees as described by Bruck et al. [2] meets these requirements: Each participant in the multicast receives a set of subordinate participants along with the multicast message. One half of this set is delegated to the next subordinate participant and this is repeated until the set is empty. The implementation reuses the existing multicast subsystem and its task messages. Because receiver cores shall propagate the message further, they need additional information about the ID range of subordinate cores and the own predecessor. The predecessor is used to aggregate the acknowledgments towards the root of the tree, similar to software tree combining [8].

The original membarrier implementation, called "private expedited", scans through all cores in order to find the cores that need to receive the multicast because they execute a thread of the caller's process. However, we discovered that this is not necessary because Linux already keeps track of this information in a cpu mask for each address space. The evaluation covers the original and simplified variant.

3 Evaluation

The evaluation was conducted at the HPI Future SOC Lab on a 4-socket machine with 18-core Intel Xeon CPU E7-8890 v3 processors (2.50 GHz), which provides 72 physical cores or 144 hardware threads in total.

The Linux kernel version 4.16.7 was extended with the new multicast implementation. The modified kernel is evaluated bare-metal without virtualization in order to exclude noise from hypervisors and para-virtualization. Throughout the evaluation, all threads are pinned to a dedicated core in the following order: The threads are first assigned to the first hardware thread of each core of the first NUMA domain, then the second NUMA domain, and so on. After 72 threads have been bound to the cores, the binding repeats with the second hardware thread of each core.

The first subsection reports on the latency of individual multicasts for the following variants: The original Linux implementation of the membarrier system call (*flat+scan*), an implementation using the cpu mask instead of scanning all cores (*flat+mask*), and the on-the-fly tree implementation (*tree+mask*). In the second sub-

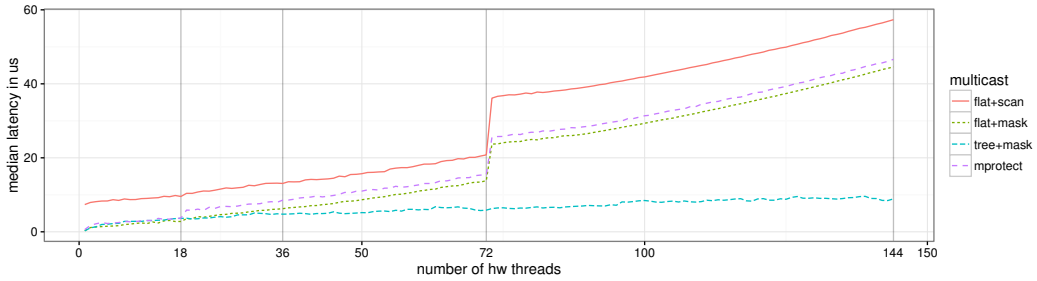


Figure 1: Median membarrier latency in microsecond for varying number of receiver threads

section, the impact on throughput is evaluated with application-level benchmarks from the uRCU library.

3.1 Multicast Latency

All threads that participate in a measurement, excluding the first thread, are running in an idle loop such that the multicast interrupt actually preempts a running application. The first thread measures the latency of a single membarrier system call. For orientation, the latency of the `mprotect` system call, which performs a TLB shootdown multicast, is included. This system call uses the Linux multicast implementation and the cpu mask similar to our *flat+mask* variant.

Figure 1 shows the latency across a growing number of participating cores. For the flat variants and the unmodified mprotect TLB shootdown, the latency scales mostly linearly with the number of cores. As to be expected, the latency of the tree-based multicast *tree+mask* is roughly logarithmic. Surprisingly, crossing NUMA domains does not seem to have a notable effect. The latency increases drastically when beginning to include the second hardware thread of the cores. *tree+mask* seems to be more robust against this effect.

The former benchmark only measures the latency of a single multicast in isolation. The second benchmark utilizes three groups of threads in order to assess the impact of concurrent multicasts. Only a single thread carries out the measurements. In parallel, a variable number of threads is repeatedly invoking the membarrier system call. Like before, the remaining threads execute the idle loop.

Figure 2 shows the latency for different numbers of threads and increasing number of concurrent multicasts. The latency of the original Linux multicast implementations scales linearly with the number of concurrent multicasts. In contrast, the tree-based multicast shows a visible impact and jitter.

This can be explained as follows: In the flat variant, solely the initiator spends time for sending the multicast messages. Thus, concurrent multicasts are perfectly parallelized without any additional overhead for the tree construction. In the tree-based multicast, the effort of sending up to $O(\log n)$ messages is delegated to another core. Whenever such tasks from concurrent multicasts hit the same core, the propagation of these multicasts is delayed while their initiator is just waiting for the acknowledge-

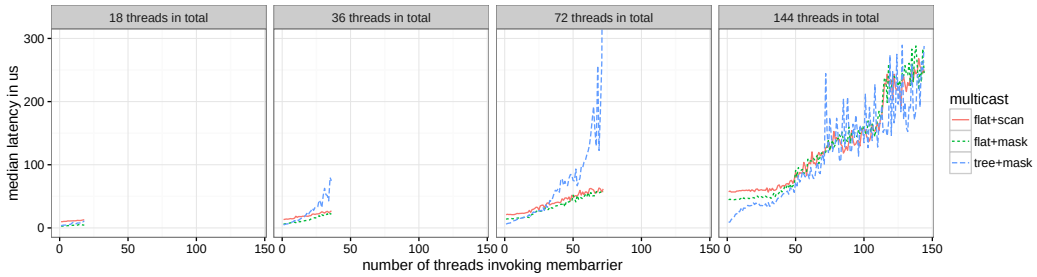


Figure 2: Median latency of concurrent membarrier calls in microseconds for a varying number of writer threads that invoke the membarrier system call

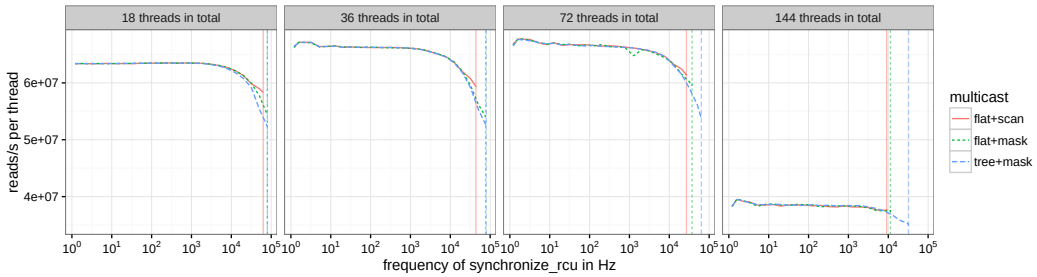


Figure 3: uRCU reader throughput with varying frequency of write operations. Vertical lines denote the maximal frequencies archived with each implementation.

ment. Even though our implementation begins the tree construction at individual offsets in the cpu mask in order to mitigate this bottleneck, the impact was strong in this micro-benchmark as can be seen with the 72-core case in Figure 2.

3.2 Impact on uRCU Throughput

This section examines the influence of the membarrier variants on the throughput of uRCU applications. To this end, two different multi-threaded throughput micro-benchmarks using the uRCU library have been implemented. The first benchmark examines the impact of a single writer on reader the throughput. The second benchmark examines the throughput with a mix of read and write operations.

Fixed frequency. The first benchmark has a single writer thread that calls `synchronize_rcu` with a configurable frequency. It simulates a system where reads dominate the workload, but might be disturbed by the multicast implementation.

Figure 3 shows the read operation throughput for varying write frequencies. Up to 10kHz, there is only a minor impact ($< 1.25\,\%$) on reader throughput compared to the conventional implementation of membarrier. Beyond 10kHz, the reader throughput of both alternative implementations drops slightly faster than for the default implementation. At the maximal frequency of standard implementation the reader throughput is decreased by less than 10 %. Because the tree-based implementation

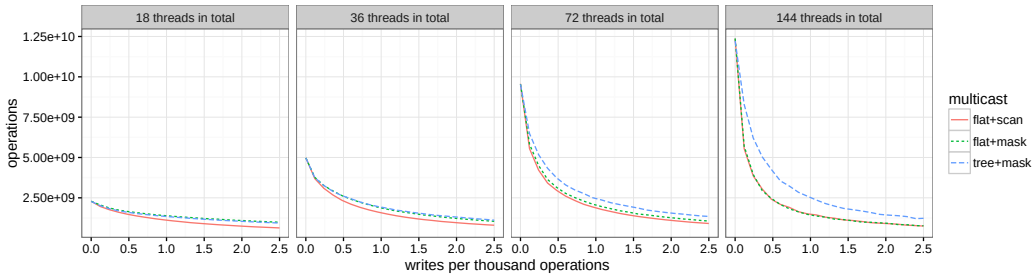


Figure 4: uRCU throughput with varying write to read ratio

Figure 5: uRCU throughput with varying write to read ratio normalized to *flat+scan*

of membarrier has a much smaller latency at large thread counts, a major increase in maximal frequency of `synchronize_rcu` can be observed. However, even at the maximal frequency for tree-based multicast, the reader throughput is decreased by less than 15 %.

Operation mix. This benchmark uses a configurable ratio between read and write operations in the range from 0‰ to 2.5‰ write operations. It represents a system under full load with a rate of internally generated write events. The read throughput decreases rapidly when the ratio of write operations raises because the membarrier in each write operation has a huge latency compared to read accesses. Even if the write operations make up a small amount of the total operations, the throughput is easely dominated by the cost of the write operations. Hence, the evaluation shows relativly small ratios of writers to readers.

For this benchmark, all waiting inside `synchronize_rcu` has been modified to use only user-space spinlocks. Otherwise, on the used benchmark machine, the hard-coded constant `URCU_WAIT_ATTEMPTS` will always lead to waiting in a Linux futex and then sleeping in 10μs intervals using the `poll` system call. The sequential futex notification of all batched threads that called `synchronize_rcu` would present a significant bottleneck with just linear scaling.

Figure 4 shows the throughput for a growing percentage of write operations and Figure 5 show the same as speedup relative to the conventional *flat+scan* variant. The tree-based variants can improve the overall throughput by a factor up to 1.48 for

72 threads and up to factor 1.77 for 144 threads. The flat implementation using the mask archives a speedup up to 1.55 for 36 threads and 1.16 for 72 threads.

When comparing the speedup it becomes obvious that the throughput is heavily influenced by the latency of membarrier. This is caused by the batching employed by the uRCU library, which restricts the number of writes per thread to a fixed rate based on the cost of the `synchronize_rcu`.

4 Conclusions

This project evaluated a shared-memory multicast mechanisms for highly dynamic multicast groups by comparison to the conventional membarrier multicast implementation in the Linux kernel.

The latency benchmarks show an improvement of the underlying multicast mechanism. Indeed, logarithmic scaling over the number of receiver cores is possible even when constructing the tree topology on the fly. In contrast to previous benchmarks on older processors, the 72-core machine of the HPI Future SOC Lab exhibits no throughput bottleneck in the processor's interrupt delivery such that the logarithmic scaling of the parallel propagation of the multicast is actually visible.

The user-space RCU micro-benchmark shows that the throughput is not inhibited much by the tree-based multicast compared to the best sequential multicast. However, it is still inconclusive if real-world applications can actually benefit from the lower latency.

Our next step is to investigate the application of the on-the-fly tree construction for the TLB invalidation multicast. This should enable a throughput improvement for applications that are sensitive to the latency of changes to the logical address space and page mapping. For example, multi-threaded out-of-core computations such as map-reduce frameworks might benefit.

Acknowledgement This work was supported by the German Research Foundation (DFG) under grant no. NO 625/7-2.

References

[1] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. "The Multikernel: A New OS Architecture for Scalable Multicore Systems". In: *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*. SOSP '09. Big Sky, Montana, USA: ACM, 2009, pages 29–44. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629579.

[2] J. Bruck, L. D. Coster, N. Dewulf, C.-T. Ho, and R. Lauwereins. "On the design and implementation of broadcast and global combine operations using the postal model". In: *IEEE Trans. Parallel Distrib. Syst.* 7.3 (Mar. 1996), pages 256–265. ISSN: 1045-9219. DOI: 10.1109/71.491579.

[3] D. Dice, M. Herlihy, and A. Kogan. "Fast Non-intrusive Memory Reclamation for Highly-concurrent Data Structures". In: *Proceedings of the 2016 ACM SIGPLAN International Symposium on Memory Management*. ISMM 2016. Santa Barbara, CA, USA: ACM, 2016, pages 36–45. ISBN: 978-1-4503-4317-6. DOI: 10.1145/2926697.2926699.

[4] S. Kaestle, R. Achermann, R. Haecki, M. Hoffmann, S. Ramos, and T. Roscoe. "Machine-aware Atomic Broadcast Trees for Multicores". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI'16. Savannah, GA, USA, 2016, pages 33–48. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/kaestle (last accessed 2020-04-01).

[5] R. M. Karp, A. Sahay, E. E. Santos, and K. E. Schauser. "Optimal Broadcast and Summation in the LogP Model". In: *Proceedings of the Fifth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '93. Velen, Germany: ACM, 1993, pages 142–153. ISBN: 0-89791-599-2. DOI: 10.1145/165231.165250.

[6] P. E. McKenney. *Memory Barriers: a Hardware View for Software Hackers*. Technical report. Beaverton: Linux Technology Center, IBM, 2010. URL: http://www.rdrop.com/users/paulmck/scalability/paper/whymb.2010.07.23a.pdf.

[7] S. Ramos and T. Hoefler. "Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL". In: *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. May 2017, pages 297–306. DOI: 10.1109/IPDPS.2017.30.

[8] P. Tang and P.-C. Yew. "Software combining algorithms for distributing hotspot addressing". In: *J. Parallel Distrib. Comput.* 10.2 (1990), pages 130–139. ISSN: 0743-7315. DOI: 10.1016/0743-7315(90)90022-H.

[9] H. Wen, J. Izraelevitz, W. Cai, H. A. Beadle, and M. L. Scott. "Interval-based Memory Reclamation". In: *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. PPoPP '18. Vienna, Austria: ACM, 2018, pages 1–13. ISBN: 978-1-4503-4982-6. DOI: 10.1145/3178487.3178488.

Assessing Certainty of QRS Detection on Single-Lead Electrocardiograms Based on Artificial Neural Networks

Jonas Chromik, Jossekin Beilharz, and Lukas Pirl

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

QRS detection in electrocardiograms (ECGs) is prerequisite for further analyses of ECG data. New technologies in ECG recording devices create new challenges for QRS detectors in terms of noise tolerance. Based on recent developments in QRS detectors based on artificial neural networks, we propose a method for assessing how certain a detection of a QRS complex is in the presence of noise.

1 Introduction

Electrocardiography (ECG) is a technique used to measure the electrical activity of the heart producing an artifact called electrocardiogram (also abbreviated ECG). ECGs are widely used for diagnosis of cardiac diseases and conditions [9, p. 26]. While some conditions are complex and need multiple ECG leads for proper diagnosis, others are to be determined algorithmically using only a single lead.

The QRS complex (see figure 1 at sample 100) is the most dominant feature of the ECG signal corresponding to the contraction of the hearts left and right ventricle [9, p. 37]. Detecting the QRS complex in an ECG signal is important in terms of determining heart rate, heart rate variability [8] and clustering or classifying heart beats according to their origin [7].

With new technologies in ECG recording devices face the problem of detecting QRS complexes in data having a low signal-to-noise ratio without the option of switching data source by using another lead. We build on top of recent developments in QRS detection based on artificial neural networks to propose a method for assessing the certainty of QRS detection. For each detected QRS complex we compare the neural networks output with an rectified version of the output. This can be used to detect and cope with noise in the signal. We demonstrate the usefulness of this method by showing how reasonable selection of a certainty threshold improves detection.

2 Challenges

With ECG-enabled smart watches [6] and ECG patch devices [1] we face the problem of QRS detection under the special conditions of:

1. ECG recordings with only a single lead since there are only two electrodes for measuring differences in electric potential (voltage) involved, and
2. Low signal-to-noise ratios since electrodes are located either
   - In smart watches: Very distal on the limbs and hence contaminated with muscle noise due to movement, or
   - In ECG patch devices: Very close to each other and hence having low differences in electric potential (voltage) and therefore low signal amplitudes.

Moody et al. [4] define 4 different sources and types of noise. *Baseline Wander* which is low frequency noise caused by motion of the subject. *Electrode Motion* which is noise in frequency range of the ECG signal caused by mechanical forces acting on the electrodes. *Muscle Movement* which is also noise in frequency range of the ECG signal but caused by muscle activity of the subject. *Power Line Interference* which is high frequency noise (usually $50Hz$, $60Hz$ or multiples of these) caused by sources of alternating current near the ECG recorder or its cables.

Baseline wander and power line interference can easily be removed by digital filters (e.g. bandpass filter with a passband between $5Hz$ and $15Hz$) [5]. Electrode motions and muscle movements are in the frequency range of the ECG signal and can hence not be removed by frequency filters. Thus we have to deal with these types of noise differently.

3 Approach

There are different approaches to algorithmic QRS detection involving techniques from digital signal processing (e.g. digital filters) or mathematical domain (e.g. Hilbert transform) [3]. The historical background of these approaches reaches back to the first implementation of a QRS detector in 1985 by Pan and Tompkins [5]. Consequently, there has been much research in this field and thus these approaches are well investigated. We want to perform our investigations in the field of ANN-based QRS detectors since great improvements in this field where made only in recent developments (compare [2] and [10]).

Project Idea We want to assess under which circumstances QRS detection becomes unreliable due to the presence of noise. For this we want to define a metric for assessing the certainty of QRS detection by an ANN-based QRS detector. Furthermore, we want to investigate how QRS detection can be improved in noise-contaminated single-channel ECG data using this metric.

State of the Art Utilizing artificial neural networks (ANNs) for QRS detection was first proposed in 1997 by [2]. Our work is based the convolutional neural network architecture proposed by [10] in 2018.

For ANN-based QRS detection, the following steps are performed:

1. Take for each sample of the ECG signal a neighborhood ("window") of n samples as representative.
2. Use the ANN to classify each window whether a QRS complex is visible on the window (class 1) or not (class 0).
3. Interpret the classification information as a new signal ("trigger signal") ideally having a square pulse shape. Parts of the trigger signal with value 1 ("plateaus") represent parts of the ECG signal with a QRS complex. However, in the presence of noise the trigger signal degenerates and thus differs from the expected square pulse shape, making it harder to find the QRS complexes.

Assessing Certainty Previous works focusses largely on preprocessing of the ECG signal and neural network architectures. Hence, we focus on postprocessing, i.e. interpreting the trigger signal in the presence of noise.

We propose assessing the certainty of each detected QRS complex by comparing the plateau found in the trigger signal with the expected ideal square pulse shape. Figure 1 shows the original (flawed) trigger signal and the square pulse shape we generated from it using a rectification step involving discretization and ripple removal.

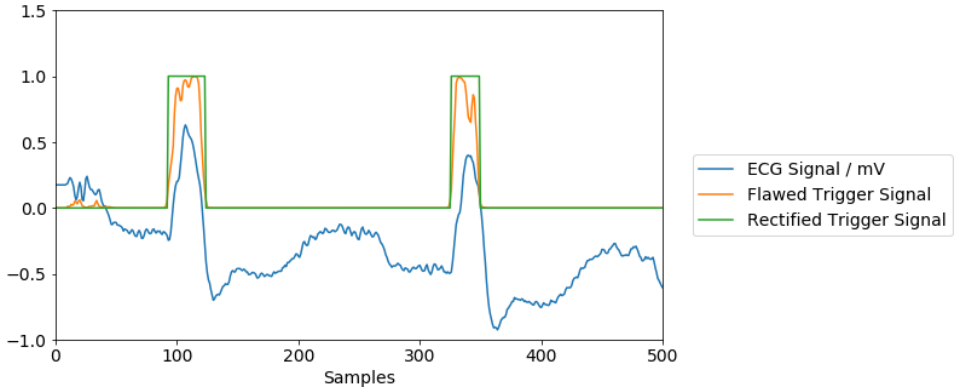


Figure 1: The original (flawed) trigger signal (orange) presents a shape that greatly differs from a square pulse shape. Using a rectification step we produced a rectified trigger signal (green) presenting a perfect square pulse shape.

Equation (1) shows the definition of certainty C_i, where i is the number of the corresponding plateau, b_i and e_i are the sample numbers where the plateau begins and ends, fts is the flawed trigger signal and rts is the rectified trigger signal.

$$C_i = \frac{\int_{b_i}^{e_i} fts(x)dx}{\int_{b_i}^{e_i} rts(x)dx} \tag{1}$$

4 Findings

We found, that we can not only use certainty to find noise-contaminated parts of the ECG signal but also to improve the performance (in terms of metrics like sensitivity, positive predictive value and F1 score) be selecting a proper certainty threshold *ct*. Detected QRS complexes with an associated certainty below *ct* are rejected while those with certainty above or equal *ct* are retained.

Suitable values for *ct* can be found by plotting the number of QRS complexes found ($\#_{QRS}$) subject to *ct*. Figure 2 shows this plot. For low *ct* there are many QRS complexes found, some of which are false positives caused by noise. For high *ct* there are few QRS complexes found and there are false negatives, i.e. QRS complexes that should have been detected but are in fact not. For intermediate *ct* there is an area of low slope where wrongly detected QRS complexes caused by noise are already removed but actual QRS complexes are still retained.

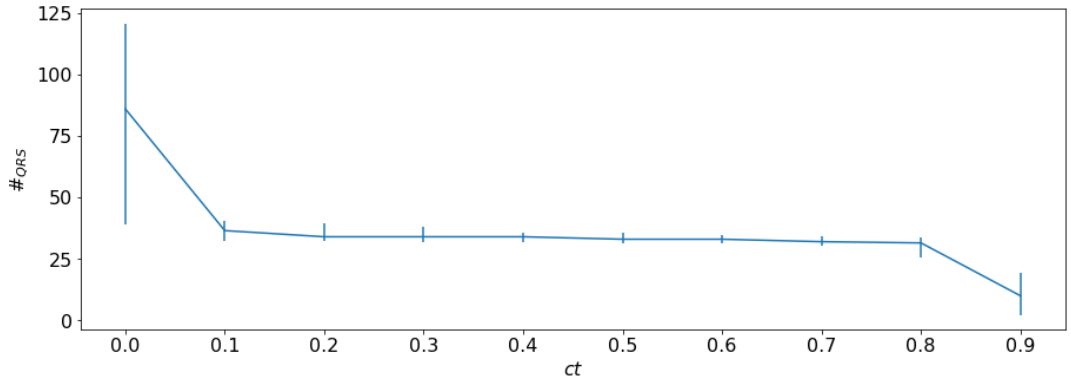


Figure 2: $\#_{QRS}$ subject to *ct* for ECG signals with SNR = 6 dB. The curve exhibits an area of low slope in the middle indicating reasonable certainty thresholds are values between 0.1 and 0.8. Lower values cause false positives (too high $\#_{QRS}$), higher values cause false negatives (too low $\#_{QRS}$).

Figure 3 shows performance metrics subject to *ct* using the same values for *ct* as in figure 2. This shows that performance, measured as F1 score which is the harmonic mean of sensitivity and positive predictive value, has its maximum value in the same *ct* range as figure 2 shows the area of low slope.

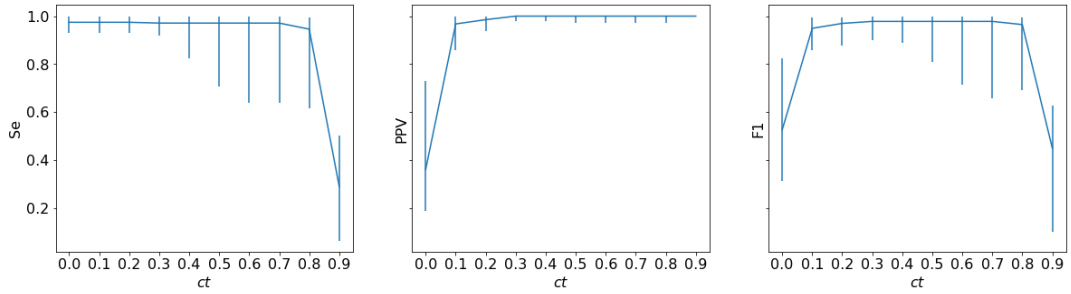


Figure 3: Sensitivity, positive predictive value and F1 score subject to *ct* for ECG signals with SNR = *6dB*. This plot shows that *ct* values producing the area of low slope in figure 2 correspond to high detection performance.

5 Used Future SOC Lab Resources

We used the GPU clusters of the Future SOC Lab for training the artificial neural networks. Software used for this is Keras[1] on Python[2] using Tensorflow[3]

6 Next Steps

Our next step is to use certainty assessment as proposed in this work for comparing the QRS detection performance in single-lead ECGs with the performance on multi-lead data to find out how and when single-lead ECG appliances become unreliable.

References

[1] E. Fung, M.-R. Järvelin, R. N. Doshi, J. S. Shinbane, S. K. Carlson, L. P. Grazette, P. M. Chang, R. S. Sangha, H. V. Huikuri, and N. S. Peters. "Electrocardiographic Patch Devices and Contemporary Wireless Cardiac Monitoring". In: *Frontiers in Physiology* 6 (May 2015). ISSN: 1664-042X. DOI: 10.3389/fphys.2015.00149.

[2] C. García-Berdonés, J. Narváez, U. Fernández, and F. Sandoval. "A New QRS Detector Based on Neural Network". In: *Biological and Artificial Computation: From Neuroscience to Technology*. Edited by J. Mira, R. Moreno-Díaz, and J. Cabestany. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pages 1260–1269. ISBN: 978-3-540-69074-0.

[1] https://keras.io/ (last accessed 2020-04-01).
[2] https://www.python.org/ (last accessed 2020-04-01).
[3] https://www.tensorflow.org/ (last accessed 2020-04-01).

[3] B. Kohler, C. Hennig, and R. Orglmeister. "The Principles of Software QRS Detection". In: *IEEE Engineering in Medicine and Biology Magazine* 21.1 (Jan. 2002), pages 42–57. ISSN: 0739-5175. DOI: 10.1109/51.993193.

[4] G. B. Moody, W. Muldrow, and R. G. Mark. "A Noise Stress Test for Arrhythmia Detectors". In: *Computers in cardiology* 11.3 (1984), pages 381–384.

[5] J. Pan and W. J. Tompkins. "A Real-Time QRS Detection Algorithm". In: *IEEE Transactions on Biomedical Engineering* BME-32.3 (Mar. 1985), pages 230–236. ISSN: 0018-9294. DOI: 10.1109/TBME.1985.325532.

[6] *Redesigned Apple Watch Series 4 Revolutionizes Communication, Fitness and Health*. Sept. 2018. URL: https://www.apple.com/newsroom/2018/09/redesigned-apple-watch-series-4-revolutionizes-communication-fitness-and-health/ (last accessed 2019-04-01).

[7] G. Sannino and G. De Pietro. "A Deep Learning Approach for ECG-Based Heartbeat Classification for Arrhythmia Detection". In: *Future Generation Computer Systems* 86 (Sept. 2018), pages 446–455. ISSN: 0167-739X. DOI: 10.1016/j.future.2018.03.057.

[8] Task Force of the European Society on Electrophysiology. "Heart Rate Variability: Standards of Measurement, Physiological Interpretation, and Clinical Use". In: *Circulation* 93.5 (Mar. 1996), pages 1043–1065. ISSN: 0009-7322, 1524-4539. DOI: 10.1161/01.CIR.93.5.1043.

[9] G. S. Wagner and D. G. Strauss. *Marriott's Practical Electrocardiography*. 12th edition. Philadelphia, PA: Lippincott Williams & Wilkins, Dec. 2013. ISBN: 978-1-4511-4625-7.

[10] Y. Xiang, Z. Lin, and J. Meng. "Automatic QRS Complex Detection Using Two-Level Convolutional Neural Network". In: *BioMedical Engineering OnLine* 17 (Jan. 2018). ISSN: 1475-925X. DOI: 10.1186/s12938-018-0441-4.

Exploring Game-Theoretic Formation of Realistic Networks

Tobias Friedrich, Pascal Lenzner, and Christopher Weyand

Algorithm Engineering Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

Many real world networks from different domains share the same structural properties. So far, there only exist models which reproduce these properties via a random process and thus have only limited explanatory power. In contrast to this, we have developed an agent-based game-theoretic model which promises a better explanation of the structure of real world networks. Our new model was investigated in computationally demanding large-scale experiments performed on the hardware of the HPI Future SOC Lab.

1 Introduction

Complex networks from the Internet to various (online) social networks have a huge impact on our lives and it is thus an important research challenge to understand these networks and the forces that shape them. The emergence of the Internet has kindled the interdisciplinary field of Network Science [3], which is devoted to analyzing and understanding real-world networks.

Extensive research, e.g. [1, 3, 4, 6, 14, 17, 18], on real world networks from many different domains like communication networks, social networks, protein-protein interaction networks, neural networks, etc. has revealed the astonishing fact that most of these networks share the following basic properties:

- *Small-world property:* The diameter and average distances in these networks are logarithmic in the number of nodes or even smaller.
- *Clustering:* Two nodes which are both adjacent to a third node have a high probability of being neighbors themselves, i.e. real networks contain an abundance of triangles and small cliques.
- *Power-law degree distribution:* In real networks the probability that a node has degree k is proportional to $k^{-\beta}$, for some constant $2 \leq \beta \leq 5$. That is, the degree distribution follows a power-law. Such networks are called *scale-free networks*.

The phenomenon that real world networks from different domains are very similar calls for a scientific explanation, i.e. formal models which generate networks with the above properties from very simple rules.

Many such models have been proposed, most prominently the preferential attachment model [4], the Chung-Lu random graph model [8], hyperbolic random graphs [13, 15] and geometric inhomogenous random graphs [5]. However, all these

models describe a purely random process which eventually outputs a network having realistic properties. On the one hand, this is desirable for sampling such networks, e.g. for testing algorithms on them, but on the other hand having a purely random process yields only a limited explanation of the structure of real world networks. Most real world networks evolved over time by the interaction of various rational agents. In case of the Internet the selfish agents correspond to the Internet Service Providers which control the Autonomous Systems, in case of social networks, the agents are people or companies who choose carefully with whom to connect. Thus, a model with higher explanatory power should consider rational selfish agents which use and modify the network to their advantage. Such models are at the core of the young field of Algorithmic Game Theory [19, 20].

2 Networks via Game Theory

In game-theoretic models for network formation selfish agents are associated to nodes of a network. Each agent chooses as strategy any subset of other agents to form a link to. The union of all links which are chosen by some player then determines the links of the created network.

The individual goal of each agent is modeled via a cost function, which typically consists of costs for creating links and of a service cost term, which measures the perceived quality of the created network for the individual agent, e.g. the service cost could be the sum of distances to all other agents [12] or just the number of reachable agents [2].

Any assignment of strategies to agents is considered an outcome of the game. Among all those outcomes the so-called equilibria are particularly interesting. In an equilibrium no agent wants to change her current strategy, given that all other players' strategies are fixed, i.e. no agent can reduce her costs in the current situation by forming another set of links. Analyzing the structure of such equilibrium networks then ideally yields insights into why real world networks exhibit the mentioned properties.

So far, such game-theoretic approaches can explain the small-world property, that is, it was proven that the diameter of all equilibrium networks is small [11]. However, to the best of our knowledge, no known game-theoretic model can explain the emergence of clustering and a power-law degree distribution. Thus, it is still an open problem to find and validate such a model.

3 Aims of the Project

Building on our previous work [7, 10, 16], we have developed a new game-theoretic model, called *strategic network augmentation*, which promises to solve the open problem. Initial experiments performed on the Future SOC Lab compute cluster [21, 22] revealed that the obtained equilibrium networks from our new model have the small-

world property, show significant clustering and the node degree distribution seems to be governed by a power-law. Moreover, our experiments established that certain minimum number of nodes is needed to reliably generate the desired properties. It turned out that experiments with $n = 1000$ nodes are rather inconclusive but that experiments with $n \geq 20000$ nodes yield valuable insights. This raised the question of reinvestigating the parameter space of the model for $n \geq 20000$ nodes.

As first step towards this goal we wanted to evaluate a core ingredient of our game-theoretic model: the behavior of the selfish agents, in particular, whether in any step of the process the active selfish agents selects his best possible improving strategy, or simply the first improving strategy found via greedy local search. For this, we wanted to compare the outcomes of both processes with respect to various properties such as the obtained power-law exponent, the obtained clustering, the number of edges in the created networks and the number of rounds until the process converged.

4 Used Future SOC Lab Resources

The experiments were run on the high-performance cluster of the HPI Future SOC Lab. The cluster consisted of 22 nodes with 80 cores each and 1TB of memory each.[1] The experiments were run via the slurm job scheduler on all nodes in parallel.

In total we generated over 10000 networks with 1000 to 100000 nodes and extracted various properties along the way to monitor the process. Later, we analyzed the generated networks using various metrics.

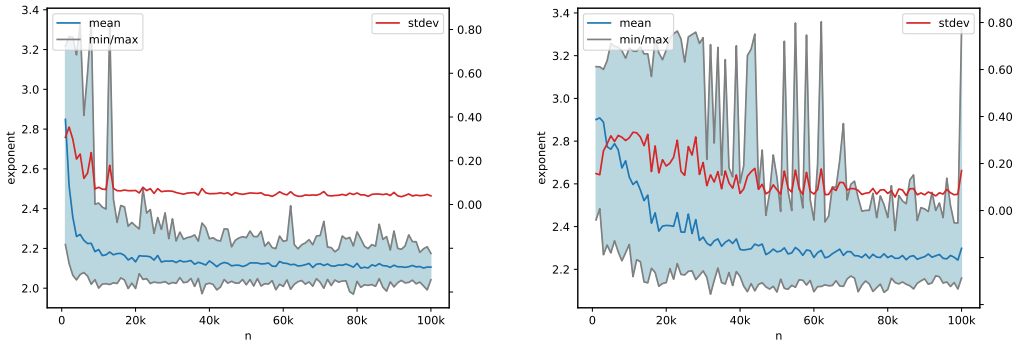
5 Findings

The experiments revealed that both variants — greedy strategy choice and best possible strategy choice — reliably generated a power-law exponent between 2.1 and 2.6. However, in the greedy version the results are much more concentrated around exponent 2.2 (See figure 1).

Regarding the average local clustering coefficient, the difference between greedy and best possible strategy choice seems insignificant. This is interesting, since we expected a higher clustering coefficient in the best strategy version. However, the experiments revealed a tendency towards the latter, i.e., the mean average local clustering coefficient is slightly higher in the best strategy version compared to the greedy strategy version (See figure 2).

As expected, the number of rounds until the process converged to an equilibrium network turned out to be lower in the best strategy version compared to greedy strategy choice. Our experiments revealed that best strategy choice is roughly 25 % faster

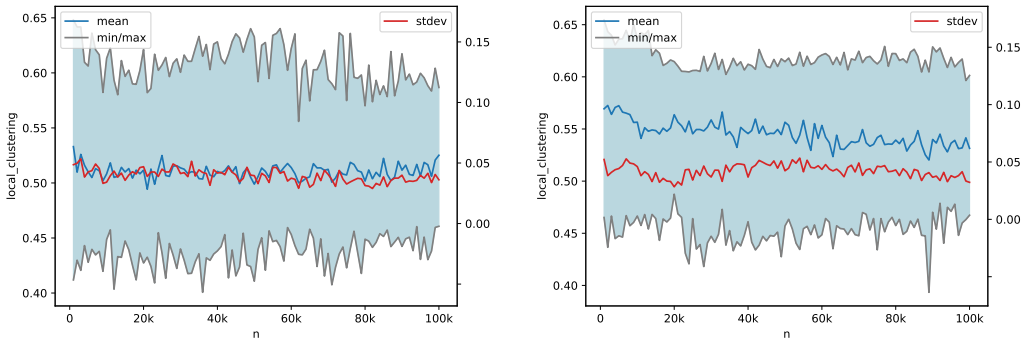
[1] Further hardware specifics can be found here: https://hpi.de/en/research/future-soc-lab/equipment.html (last accessed 2020-04-01)



(**a**) Measured power-law exponent with greedy strategy choice

(**b**) Measured power-law exponent with best possible strategy choice

Figure 1: Comparison of the obtained power-law exponents

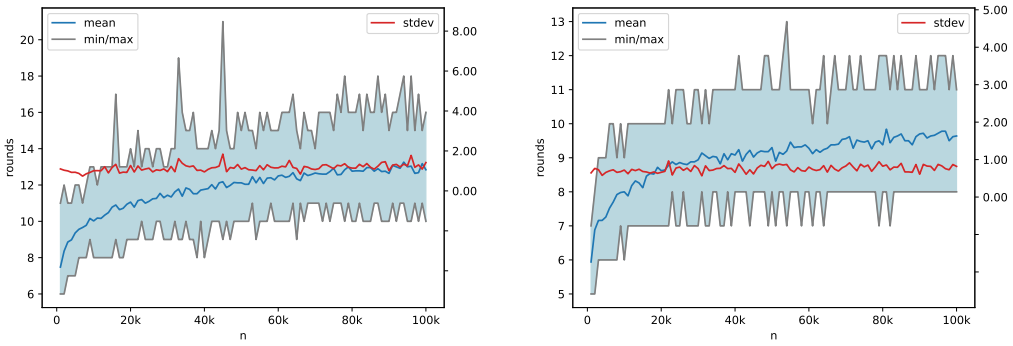


(**a**) Measured clustering with greedy strategy choice

(**b**) Measured clustering with best possible strategy choice

Figure 2: Comparison of the obtained average local clustering coefficient

than greedy strategy choice. This is interesting, since each strategy change in the best strategy version is computationally much more demanding. In total computation time, it turned out that the best strategy version takes more computation time until convergence compared to the greedy version.



(**a**) Number of rounds until convergence with greedy strategy choice

(**b**) Number of rounds until convergence with best possible strategy choice

Figure 3: Comparison of the convergence speed

6 Next Steps

We made the first step towards a systematic reinvestigation of the relationship between specific parameter settings and obtained network properties. As next steps, we want to reconsider different parameter choices for influencing the achieved power-law exponent and the clustering. Moreover, we expect that the obtained clustering can also be influenced by a modified strategy choice of the agents.

We have also observed other realistic properties of the equilibrium networks generated by strategic network augmentation. These need to be further investigated. One example of an additional realistic property is the appearance of a so-called *rich club* [9], which means that the high degree nodes form a connected subgraph. Another example is that our generated networks seem to be resilient against node or edge failure, also commonly observed in real networks. Here it would be interesting to investigate the influence of greedy versus best strategy choices on these properties.

Last but not least, we want to validate our models with data from real networks. That is, we want to measure if real networks are close to being in equilibrium in our setting and we want to use time series of real networks to investigate if our models predict the right structural changes over time.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. "Internet: Diameter of the world-wide web". In: *nature* 401.6749 (1999), page 130.

[2] V. Bala and S. Goyal. "A noncooperative model of network formation". In: *Econometrica* 68.5 (2000), pages 1181–1229.

[3] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[4] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pages 509–512.

[5] K. Bringmann, R. Keusch, and J. Lengler. *Geometric inhomogeneous random graphs*. preprint. 2015. arXiv: 1511.00576 [cs.SI].

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. "Graph structure in the Web". In: *Computer Networks* 33.1 (2000), pages 309–320.

[7] A. Chauhan, P. Lenzner, A. Melnichenko, and L. Molitor. "Selfish Network Creation with Non-Uniform Edge Cost". In: *SAGT'17*. Springer. 2017, pages 160–172.

[8] F. Chung and L. Lu. "The average distances in random graphs with given expected degrees". In: *PNAS* 99.25 (2002), pages 15879–15882.

[9] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. "Detecting rich-club ordering in complex networks". In: *Nature physics* 2.2 (2006), page 110.

[10] A. Cord-Landwehr and P. Lenzner. "Network Creation Games: Think Global - Act Local". In: *MFCS'15*. 2015, pages 248–260.

[11] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. "The Price of Anarchy in Network Creation Games". In: *ACM Transactions on Algorithms* 8.2 (2012), page 13.

[12] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. "On a Network Creation Game". In: PODC'03. Boston, Massachusetts: ACM, 2003, pages 347–351.

[13] T. Friedrich and A. Krohmer. "On the diameter of hyperbolic random graphs". In: *ICALP'15*. Springer. 2015, pages 614–625.

[14] J. Kleinberg. "The Small-world Phenomenon: An Algorithmic Perspective". In: *STOC'00*. STOC '00. Portland, Oregon, USA: ACM, 2000, pages 163–170. ISBN: 1-58113-184-4.

[15] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. "Hyperbolic geometry of complex networks". In: *Phys. Rev. E* 82 (3 Sept. 2010), page 036106.

[16] P. Lenzner. "Greedy Selfish Network Creation". In: *WINE'12*. 2012, pages 142–155.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *SIGKDD'05*. ACM. 2005, pages 177–187.

[18] M. Newman, A.-L. Barabasi, and D. J. Watts. *The structure and dynamics of networks*. Princeton University Press, 2011.

[19] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[20] C. H. Papadimitriou. "Algorithms, games, and the internet". In: *STOC'01*. 2001, pages 749–753.

[21] D. N. Schumann. "Exploring Game Theoretic Models for Generating Real World Networks". Master's thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[22] C. Weyand. "Locality in Game Theoretic Models for Real-World Networks". Master's thesis. Hasso Plattner Institute, University of Potsdam, 2018.

Energy Efficiency, Virtualization and Performance
Second (WEEVILS)

Carlos Juiz and Belen Bermejo

Computer Science Department
University of the Balearic Islands (Spain)
{cjuiz,belen.bermejo}@uib.es

1 Project idea

The prevailing international scientific opinion on climate change is that human activities resulted in substantial global warming from mid-20^{th} century, and that continued growth in greenhouse gas concentrations, caused by human-induced emissions, is mainly (direct or indirectly) due to energy consumption. The sector where energy consumption has grown tremendously is IT (Information Technologies). On one hand, datacentres that host cloud services are becoming huge warehouses with lot of servers, additional electronic equipment and complex cooling systems. These computers and telecommunications networks are today primarily responsible for electronical energy consumption caused by datacentres, which maintain the quality of Internet service in operation. Giving more capacity to these datacentres usually means more cloud servers and consequently additional more equipment and cooling are needed, too. On the other hand, the increasing number of users, especially due to the popularization of cloud services, produced continuously capacity problems at datacentres due to performance requirements [3, 4]. Thus, a new challenge directly related to this research area emerges: the energy efficiency of cloud servers. WEEVILS project is aimed by this combination of these topics. Our main project objectives are:

1. The characterization of the performance and energy consumption from consolidated servers through benchmarking and monitoring techniques.
2. Modelling the energy consumption patterns and performance of consolidated servers. We shall infer models of such energy, performance and virtualization, together.
3. Performing several comparative studies of benchmarking between physical and virtual servers.
4. The validation of our performance-oriented previous findings and energy efficiency virtualization models through experimentations in real datacenters, as HPI Future SOC Lab.

The WEEVILS project is defined as the extension of the previous one (WEEVIL) developed using the HPI Future SOC Lab infrastructure (RX600S5-1 server).

Then, the research question we attempt to answer in this project extension is: "How to determine the performance degradation of physical servers due to Virtual Machine Consolidation?"

2 Used Future SOC Lab resources

In order to answer the research question, we design a methodology based in the following stages:

1. To characterize the performance of physical servers through benchmarking and monitoring techniques [5].
2. Identify the sources of virtual machine consolidation overhead.
3. Measuring and quantifying the virtual machine consolidation overhead.
4. Discussing and analysing the results.

The exposed methodology was performed in the HPI Future SOC Lab infrastructure, specifically in the rx600s5-1 server which hardware has 48 CPUs and 1024 GB of RAM memory. In table 1 we can see the actions developed in each stage and the correspondent outcome.

Table 1: Actions developed in each stage and its outcomes

Stage	**Action**	**Outcome**
Stage 1	Characterizing the performance of PM through benchmarking and monitoring techniques	PM's performance when consolidating virtual machines
Stage 2	Identification of overhead sources	Description of virtual machine consolidation overheads
Stage 3	Measuring and quantifying the virtual machine consolidation overhead in HPI infrastructure and UIB's servers	Overheads' quantification
Stage 4	Discussion and results' analysis	Comparison between UIB and HPI servers in terms of overheads

3 Findings

In previous section we explain the stages we perform to answer the research question, as well as the developed actions and its outcomes. We [2] define the virtual machine consolidation overhead as the extra work that the system has to perform in order to manage the consolidated virtual machines. As a consequence, we identified two sources of overhead. The former is coming from the fact of having an hypervisor (OV_v), and the later is from the fact of having more than one allocated virtual machine (OV_c).

In figure 1, figure 2, figure 3, and figure 4 the graphical representation of the overheads' sources are represented. We can observe that the HPI infrastructure has less overhead due to the consolidated virtual machines due to the amount of available resources. On the contrary, the hypervisor installed on the HPI infrastructure needs to manage more resources, and as a consequence, the overhead due to this fact is higher than the T430 server one (from the UIB) [1].

Publications

The use of the HPI infrastructure collaborates to develop the following research works:

- B. Bermejo, C. Juiz, and C Guerrero. "On the Linearity of Performance and Energy at Virtual Machine Consolidation: the CiS2 Index for CPU Workload in Server Saturation". In: Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications. Exeter, UK, 2018, pages 928– 933.
- B. Bermejo, C. Juiz, and C. Guerrero. "Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance". In: The Journal of Supercomputing (2018). ISSN: 1573-0484. https://doi.org/10.1007/s11227-018-2613-1.
- B. Bermejo, C. Juiz, and N. Thomas, "On the virtualization overhead and energy consumption in consolidated servers." In: *UK- Performance Engineering Workshop (UKPEW)*. Newcastle upon Tyne, UK, 2018.

4 Next steps

In this project we attempt to answer the research question: "How to determine the performance degradation of physical servers due to Virtual Machine Consolidation?". For that, we design a methodology based on monitoring and benchmarking techniques and we apply it to our servers and then, we extend the experiment to HPI infrastructure.

The HPI allows us to extend the work and also to obtain more accurate results and conclusions. Due to that, there are very interesting research questions that we

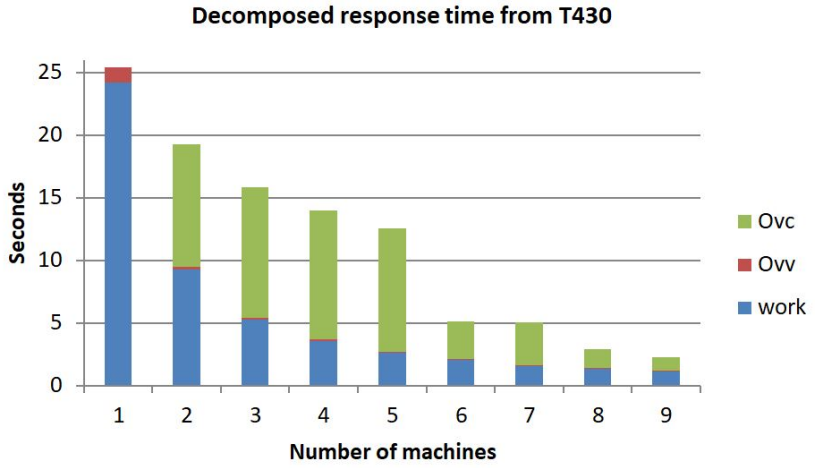


Figure 1: Descomposed response time for T430 server

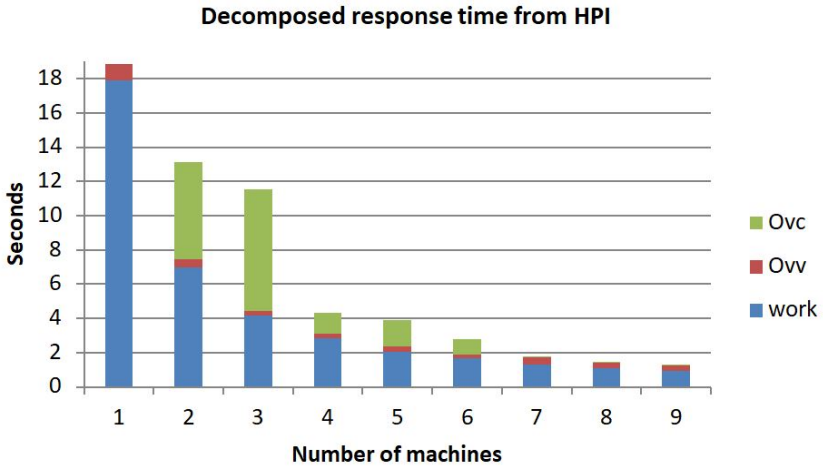


Figure 2: Descomposed response time for HPI infrastructure

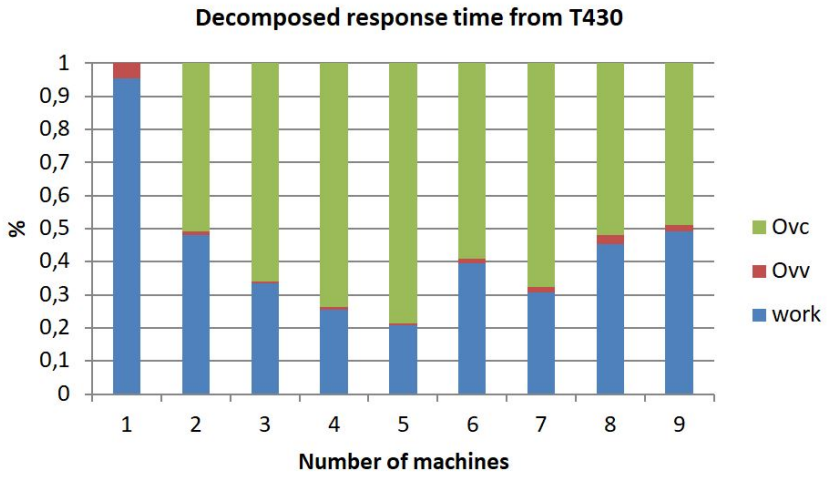


Figure 3: Descomposed response time for T430 server (in %)

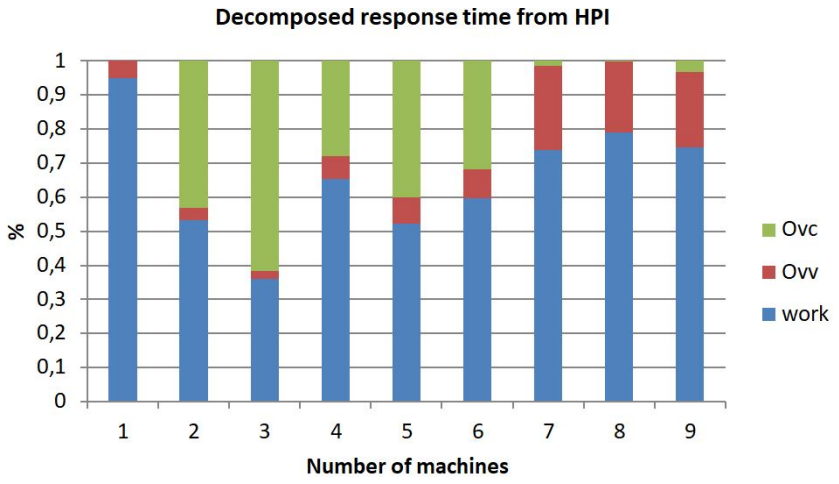


Figure 4: Descomposed response time for HPI infrastructure (in %)

would like to answer with the support to the HPI infrastructure. The questions are the following:

- How the behaviour of OV_v and OV_c under different hypervisors is?
- Could be generalize the behaviour of the virtual machine consolidation overhead?

In order to answer the proposed questions, we will apply for a new project in the Hasso Plattner Institute in order to use the HPI Future SOC Lab's IT infrastructure.

References

[1] B. Bermejo, C. Juiz, and C. Guerrero. "On the Linearity of Performance and Energy at Virtual Machine Consolidation: the CiS2 Index for CPU Workload in Server Saturation". In: *Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications*. Exeter, UK, 2018, pages 928–933.

[2] B. Bermejo, C. Juiz, and N. Thomas. "On the virtualization overhead and energy consumption in consolidated servers". In: *UK- Performance Engineering Workshop (UKPEW)*. Newcastle upon Tyne, UK, 2018.

[3] B. Bermejo, S. Filiposka, C. Juiz, B. Gómez, and C. Guerrero. "Improving the Energy Efficiency in Cloud Computing Data Centres Through Resource Allocation Techniques". In: *Research Advances in Cloud Computing*. Edited by S. Chaudhary, G. Somani, and R. Buyya. Singapore: Springer Singapore, 2017, pages 211–236. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_9.

[4] R. Buyya, C. Vecchiola, and S. T. Selvi. *Mastering Cloud Computing: Foundations and Applications Programming*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 978-0-12-409539-7.

[5] X. Molero, C. Juiz, and M. Rodeño. *Evaluación y Modelado del Rendimiento del os Sistemas Informáticos*. Pearson, 2004. DOI: 10.2200/S00516ED2V01Y201306CAC024.

The Generative Adversarial Network usage in the classification problems

The usage of GAN for Musical Genre Classification

Joao Sauer and Leandro dos Santos Coelh

Universidade Federal do Paraná
Electrical Engineering Sector
joao.sauer@gmail.com, lscoelho2009@gmail.com

Nowadays, the musical genre can be very ample and in some cases, creates discussions about what a person would consider it as only one particular genre. This kind of classification makes machine learning algorithms struggle to analyse it correctly. However, some new technologies like Generative Algorithm Networks could be manipulated and used as a classification tool to try to solve problem. Helping to identify correctly a particular kind of music focusing only in identifying the most important genre of a music.

1 Introduction

The musical field has different information that can be extracted and an interesting field to be analysed [4]. However, its features are complex for a classification tool as one feature can be common between two or more genres. Different approaches have being used, such as the usage of sentiment analyses of the lyrics [10]. In this work, we suggest some modifications in the Generative Algorithm Networks in order to allow it to be used as a classification tool and identify the major musical genre from a song.

Contributions

The contributions expected from this work is the usage of Generative adversarial network (GAN) [5] to be used as classification tool for the identification of the main musical genre from a song. This research combines two musical datasets The Million Song Dataset (MSD) [1] and Last.fm dataset [9]. The MSD contains features extracted from one million songs. The Last.fm dataset gives the genre of the music based on the opinion from humans. In addition, this research compares K-Means [3], T-SNE [11], SVM [2] and Random Forest [6] and analyse the results from each of these methods.

2 Context

Music genre classification have already being studied in the past using different approaches. One of the approaches uses sentiment analyses from the lyrics of the musics [4]. Another approach to classify the musical genre is the usage of random forest [7] or K-Means [8]. The results from all these studies proved it is possible to use an algorithm to solve this issue. However, there are some newer approaches that could be used on these datasets and it opens the opportunity to verify the quality of them against these datasets. Also, these datasets normally do not contain the audio of the music, just information about the songs and their metadata, which makes even more difficult for the classification.

3 Problem

The musical genre classification is a complex problem, once that different individuals can have distinct opinions about the same music. An individual can consider a music as *Pop-Rock* while another would consider it as only *Rock*. However, in most cases the chosen genres are co-related, which allows the usage of one common genre to be the primary. Another difficulty is the similarity between the features identified in the dataset. These similarities can mislead the correct classification of a song.

Specific Problem

In the MSD, there is no audio to be analysed. This research uses only the information provided by the dataset. One key piece of information is missing from the MSD, the genre. This information is provided by the last.FM dataset. Together, both datasets generate a new dataset that can be analysed. The merging of these datasets provides extra information from a single song such as its beats, bars, tatums, danceability, loudness and many other information.

4 Solution

Machine Learning can be used to solve this problem. It is possible to train the system to learn to identify a particular genre from a song. There are several approaches that can be used such as GAN, random forest, K-Means. This research applies some of them, makes a comparison between them and suggests a new approach. This research uses GAN since this type of network tries to verify if an entry from the dataset is part or not of a particular group of individuals. This research modifies the original GAN with clusterization of data. As a classification tool, this research uses K-Means.

Specific Solution

The GAN is an unsupervised learning method for neural networks. GAN normally is used to generate and analyse clusters of data. In this research the method was changed to support classification.

5 Implementation

This research, is on its initial stages. In this initial phase, the docker image was created with the scikit and Jupyter Notebooks. The following activities have been performed in the docker image:

- Dataset Manipulation;
- Usage of K-Means;
- Usage of Random Forest;
- Usage of GAN;

All of these activities were executed at same time as the research is on its initial phase. As the research matures, some of the activities will be used for analysis only.

5.1 Dataset Manipulation

The MSD contains a million files separated in folders by the music name. Each file is a .h5 that contains all the information related each music. Most of the them contains all the possible information. The first step was read all files with all possible valid features in a unique file that could be used in Jupyter Notebook. In addition, the original files does not contain the musical genre. This information is in a separate dataset. The key between both datasets is an ID. As a first test, only the `beats_confidence` of the files was used to try to classify the songs in the different possible genres.

5.2 K-Means

K-Means splits the dataset into genres and then map the different clusters with the most common genre found to confirm that the group found was an specific genre. Because it's an unsupervised learning, this could give the ability to give multiples genres, depending on the distance of the centroids found.

5.3 Random Forest

Random Forest, as the name suggest, is a tree creator that tries to use the features found as parameters to decide were and how deep the entry should go in the tree. With the training dataset, the tree is created and then a new test entry is passed. It

decides which node in the tree belongs and returns its result. This is a supervised method and because of this, the results are unique. Together with Random Forest, two other methods, from the same package, were used: SVM and Kernel SVM. The expectation would be to use them as a comparative for the Random Forest method.

5.4 GAN

GAN is an unsupervised learning method for neural networks. The model consists of two networks. A generator that generates the data from a noise vector, and a discriminator that discriminates between a generated data and a real data. The interesting part is that once the network is trained it can generate the data from random noise that can be used to identify the correct genre for that particular data.

6 Evaluation

Because the project has just started the evaluation was not yet used. It was implemented using the default options from scikit-learn, as shown in listing 1.

Listing 1: Code used for evaluation

```
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
print("f1_score:",metrics.f1_score(Y_test, Y_pred))
print("recall_score:",metrics.recall_score(Y_test, Y_pred))
print("precision_score:",metrics.precision_score(Y_test, Y_pred))
```

The results can be looked at table 1. The GAN network was not evaluated at this stage of the research.

Table 1: Results Evaluation

	K-Means	Random Forest	SVC	Kernel SVC
accuracy	0.056	0.529	0.338	0.519
f1_score	0.056	0.529	0.338	0.519
recall_score	0.056	0.529	0.338	0.519
precision_score	0.056	0.529	0.338	0.519

Another important point to be noticed was the usage of the confusion matrix against the random forest tests. This can be seen in figure 1 below:

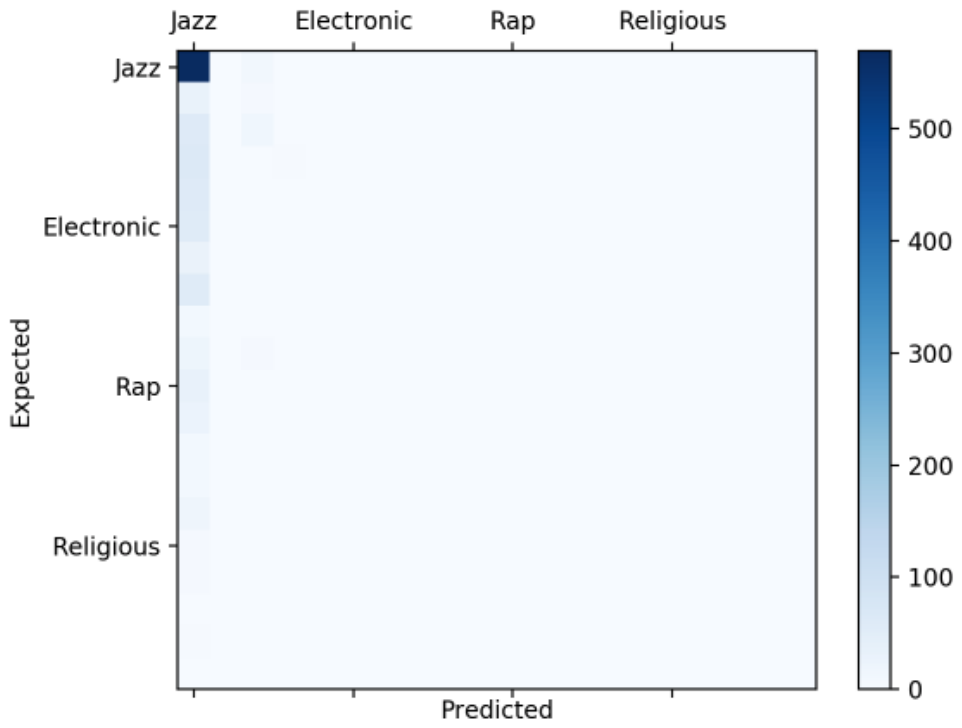


Figure 1: Confusion Matrix for Random Forest

7 Conclusion

As explained before, this work has just started. The development is still a work in progress. There are several work to be done, as showed by the results obtained so far. The dataset features are not the best ones to be used right now and its necessary to have a better approach of selecting them. On the other hand, it's possible to say that the GAN can be changed to fit the proposed work.

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[2] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pages 273–297.

[3] E. Forgey. "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification". In: *Biometrics* 21.3 (1965), pages 768–769.

[4] D. Gärtner. "Tempo Detection of Urban Music Using Tatum Grid Non Negative Matrix Factorization". In: *ISMIR*. 2013.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pages 2672–2680.

[6] T. K. Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Volume 1. IEEE. 1995, pages 278–282.

[7] X. Jin and R. Bie. "Random Forest and PCA for Self-Organizing Maps based Automatic Music Genre Discrimination." In: *DMIN*. 2006, pages 414–417.

[8] D. Kim, K.-s. Kim, K.-H. Park, J.-H. Lee, and K. M. Lee. "A music recommendation system with a dynamic K-means clustering algorithm". In: *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE. 2007, pages 399–403.

[9] *Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset*. 2011. URL: http://labrosa.ee.columbia.edu/millionsong/lastfm (last accessed 2019-02-07).

[10] D. Liang, H. Gu, and B. T. O'Connor. *Music Genre Classification with the Million Song Dataset*. Technical report 15-826 Final Report. Carnegie Mellon University, Dec. 3, 2011.

[11] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.86 (2008), pages 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

Towards GOLF: Graph Object Log Format

Graph-based Normalization within SIEMs

Pejman Najafi[1], Wenzel Pünter[2], Feng Cheng[1], and Christoph Meinel[1]

[1] Hasso Plattner Institute
{pejman.najafi,feng.cheng,christoph.meinel}@hpi.de
[2] University of Potsdam
wenzel.puenter@student.hpi.uni-potsdam.de

In this paper we introduce the Graph Object Log Format (GOLF) designed to tackle the normalization of event logs into serializable sub-graphs particualrly in the context of Security Information Event Management (SIEM) system where the underlying data is expected to be highly connected and correlated.

1 Introduction

SANS Institute identified the completeness of normalized events, correlation of log data and the normalization of events as challenging problems of traditional SIEM systems in its 9th log management survey report [12].

Normalization is the process of transforming heterogeneous data sources and formats into a standardized representation. In the field of security, normalization is the processing of collected log files from multiple network devices or services on a host into a unified representation suitable for signature matching and correlation [10].

Jaeger et al. [4] identifies rule matching as the most common normalization method and suggested the idea of hierarchical parsing. Rainer Gerhards supports this observation and suggests radix tries as an alternative normalization method with better performance characteristics.

2 Graph-based Normalization

Events are commonly represented as rows in a data frame. While this representation is used to create index structures, group events by properties and do machine learning on derived attributes, tables lack a way of natively expressing correlation. Due to this, it is often required to perform complex joins on the tables. This is of special importance when event and OSINT/TI data should be combined.

Some log formats have moved to hierarchical data formats, where the data structure is represented as a key-value tree. While hierarchical formats allow the encapsulation of objects, only parent-children relationships can be expressed. For log events, this would require additional redundancy as correlated events would need to embed their counterparts in many-to-many relationships.

An alternative to tabular- and tree-based data formats are graphs. Graphs can natively express the relationships between different entities. In the world of log data, especially the analysis of network-related events between different actors has the potential to benefit from a graph-based representation. To capture attributes of the viewed entities, one can create a property graph. This type of graph adds key-value pairs to nodes or edges.

From Events to Relationships

Many applications create log files containing references to other services. Furthermore one could see an event as list of entities with particular relationship, e.g. a successful authentication event could be represented as two entities with an edge, i.e., a user authenticating with a system.

In consequence, it is possible to model events as subgraphs. Normalization systems could incorporate a postprocessing step, which transforms data into subgraphs as an additional output of the normalization subsystem.

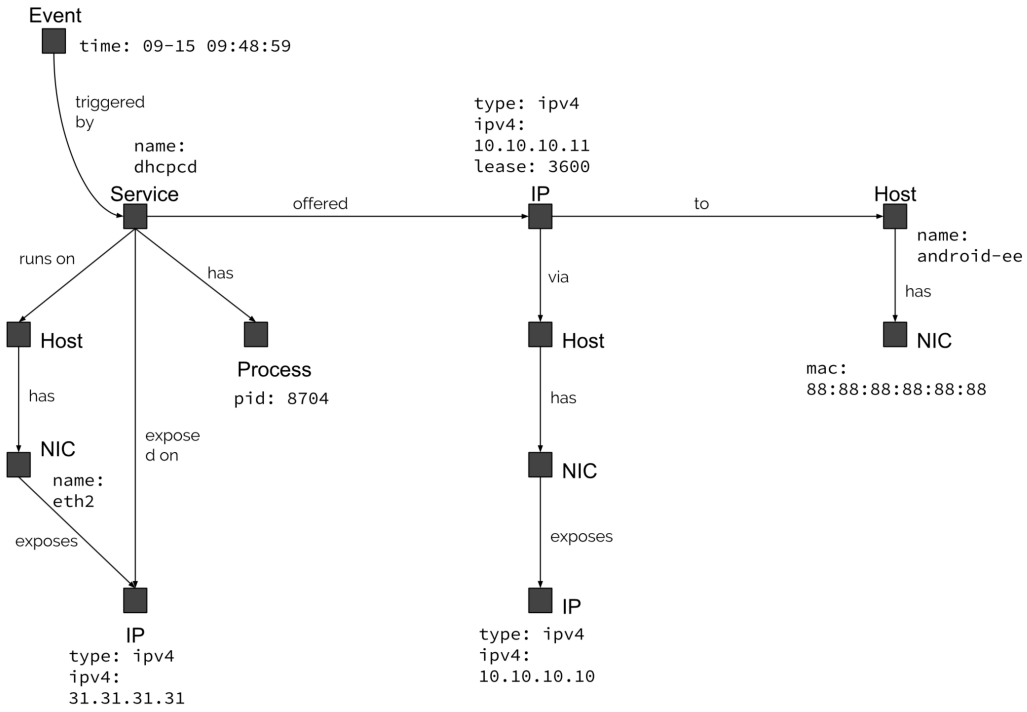


Figure 1: An example for a relationship graph

3 GOLF: Graph Object Log Format

In today's world where distributed systems and microservice architecture have replaced the majority of the traditional systems, serialization has become an important topic when it comes to efficiency and performance.

In this regard JSON is known to provide great interoperability among components within the systems, especially with NoSQL databases like Elasticsearch. Some of the most widely used JSON serializers are The popular C++ implementation by Niels Lohmann [6] and RapidJSON. [14] presents a benchmark of major JSON serializers.

However, even optimized JSON-based serialization requires around 20 % of the total execution time during event normalization, indicating a computational inefficiency of this used format. This CPU utilization is not only required in the initial step in the pipeline but for each service, which needs to decode the input, process it and then encode it again. In addition, JSON is well-known to be an inefficient format in terms of storage compared to binary representations — especially numbers are encoded in a very inefficient way, due to the use of the numeric UTF-8 representation. While graphs can be expressed as JSON using the GraphSON format [13], this representation might not fulfill the requirements of a big data platform.

Many file formats try to resolve the issues of the time and space requirements. Major competitors to JSON include XML (respective GraphML), Protobuf, and Avro. Maeda analyzed the performance of object serialization libraries in [7] and identified Protobuf as the best tradeoff between object size, serialization and deserialization time. While Avro provided the smallest serialization objects, its performance for en- and decoding has been in the middle of the compared formats.

As an alternative to these existing and standardized formats, this thesis suggests a time- and space-efficient format, that is dedicated to the problem domain of log event data, called GOLF — the Graph Object Log Format. As with e.g. Protobuf and Avro, the format should benefit from a pre-shared data model and is encoded in binary. As only subgraphs need to be transmitted, it is possible to build the system around nodes, edges and key-value attributes on both nodes and edges. Frequently, application-specific data can be represented in a more efficient way. This allows the encoding of e.g. IP addresses as 32-bit (IPv4) or 64-bit (IPv6) integers or ports as 16-bit integers. These binary representations only require a few bytes instead of one byte per digit or character and should be added in a modular way.

It should be possible to add additional types of values. The data model can be specified by the possible types of nodes, edges and attribute keys. Later steps in the processing pipeline might integrate the event subgraph into a greater context. For this, a hash-based correlation of nodes and edges might require the introduction of unique identifiers. In addition to that, time ranges might be added as a star-schema-modeled node hierarchy, attached to the corresponding nodes of a subgraph.

GOLF is based on a binary format, which has a header containing the used revision of the format, a CRC checksum for all following fields, the timestamp of processing, a byte-encoded 128 bit UUID for the processed event and source together with an identifier of the matching chain. It is possible to specify up to 256 revisions before additional version identification fields need to be introduced. The timestamp is a 64-

bit UNIX timestamp, which allows expressing time up to the year 292, 277, 026, 596. The UUID allows deduplication in different parts of the platform. With the source and chain identifiers, it is possible to specify up to 2^{64} different log sources and 2^{32} different regular expression chains. After this header, the format specifies the number of nodes and edges transferred. It is possible to encode up to 2^{16} nodes and 2^{32} edges.

Two dynamically-sized blocks contain the node and edge payload. Nodes consist of an identifier, which specifies the type with reference to the data model and a set of key-value pairs, where each key corresponds to a data model reference ID and the value is encoded depending on the type associated with the used key. GOLF allows up to 2^{16} different node types, which limits the complexity of the data model. Each node can hold up to 2^{16} key-value pairs, which is also the limitation for the maximum number of possible keys. It is possible to store up to 2^{32} bytes per value.

The second block contains edges. As each node has its own identifier, it is possible to specify a directed edge between two node IDs. Attributes for nodes are encoded as attributes for edges. As a data model can carry different semantics for nodes and edges, it is possible to encode 2^{16} additional pairs and keys for edges.

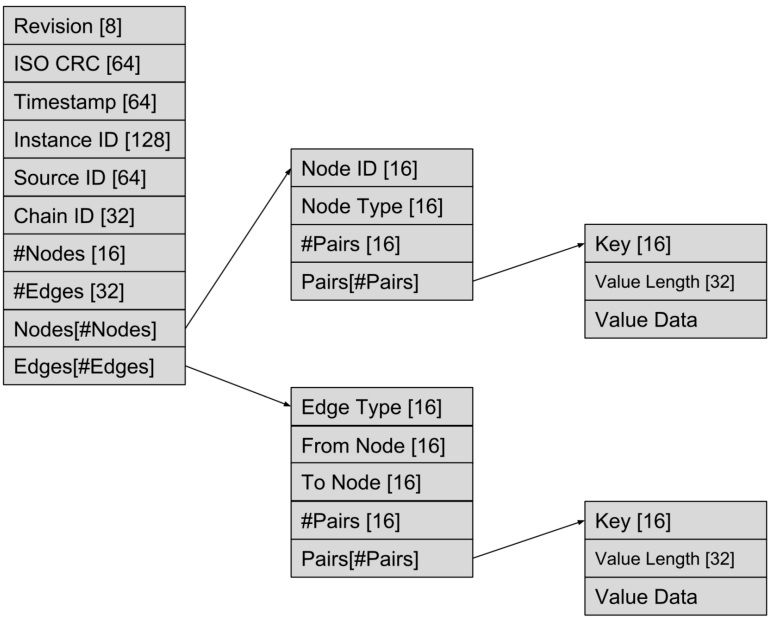


Figure 2: Memory Layout for GOLF

4 Graph-based Threat Detection

As mentioned it is important to develop an ontology of possible node and edge types along with a specification of the key and value types. This is then used to derive GOLF. Examples of such ontologies could be found at [3].

In consequence, in case of an incident, it is possible to trace the threat and its activities based on the interactions with network resources. This concept can be transferred to host systems, file and process structures [8].

Graph algorithms can operate on the discovered infrastructure relationships and help to detect lateral movements of APT actors in corporate networks [5]. Derived network graphs can also help to analyze the impact of vulnerabilities [1]. Eberle et al. [2] showed in 2010, that a graph model of access operations allows the detection of insider threats. Najafi et al. [9] used belief propagation on entity graphs derived from log data and seeded with TI information to find potentially malicious entities based on the hypothesis of infrastructure reuse.

5 Discussion and Future Work

GOLF only provides a reduced form of extensibility due to the undefined pre-exchange of the underlying data model and the focus on event serialization. While losing the ability to express arbitrary contents, GOLF has a very small message size footprint and allows faster encoding and decoding due to the unpacking into native memory-aligned structures. As a result of this, GOLF is a special-purpose network and file format for the transmission of event subgraphs. The storage on a file system requires additional effort to provide an encapsulation format, which allows the parallel processing of the files itself along with partitioning and archival capabilities. Network-oriented applications require additional encapsulation layers like Kafka or the TCP protocol. For network-based applications, additional security mechanisms – especially encryption and authentication – are mandatory.

Table 1: GOLF Comparison

Name	Standard	C++, Python, Java	self-contained	Efficient Encoding	Custom Types	Encoding speed	Size
JSON	✔	✔	✔	✘	✘	slow	large
Graphson	✔	✔	✔	✘	✘	slow	large
Protobuf	✔	✔	✘	✔	✘	fast	medium
XML	✔	✔	✔	✘	✘	slow	large
GraphML	✔	✔	✔	✘	✘	slow	large
Avro	✔	✔	✘	✔	✘	medium	small
GOLF	✘	✘	✘	✔	✔	fast	small

Further work needs to be taken into account when evaluating the efficiency of different formats. Especially Protobuf, Avro and GOLF provide the potential to solve common bottlenecks in implementations. As Protobuf and Avro use reflection in many of their implementations, GOLF might provide a benefit for just-in-time compiled languages. In addition to that, an advanced data models and ontologies should be used to provide a true benefit for graph algorithms. For the scope of the implementations described in this work, serialization has been implemented using the hierarchical OLF [11] format without the need for graph-based serialization. Future implementations should evaluate Protobuf as a dictionary and GOLF as a graph-based format.

6 Conclusion

In this research, we investigated the applicability of graph-based normalization for event logs particularly in the context of SIEM systems. We next introduced GOLF designed to tackle the serialization issue for graph based objects. Lastly we concluded with some discussion and highlight for future work extending GOLF and graph-based normalization for more feasible implementations.

References

[1] P. Ammann, D. Wijesekera, and S. Kaushik. "Scalable, graph-based network vulnerability analysis". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM. 2002, pages 217–224.

[2] W. Eberle, J. Graves, and L. Holder. "Insider threat detection using a graph-based approach". In: *Journal of Applied Security Research* 6.1 (2010), pages 32–81.

[3] M. D. Iannacone, S. Bohn, G. Nakamura, J. Gerth, K. M. Huffer, R. A. Bridges, E. M. Ferragut, and J. R. Goodall. "Developing an Ontology for Cyber Security Knowledge Graphs." In: *CISR* 15 (2015), page 12.

[4] D. Jaeger, A. Azodi, F. Cheng, and C. Meinel. "Normalizing security events with a hierarchical knowledge base". In: *IFIP International Conference on Information Security Theory and Practice*. Springer. 2015, pages 237–248.

[5] J. R. Johnson and E. A. Hogan. "A graph analytic metric for mitigating advanced persistent threat". In: *2013 IEEE International Conference on Intelligence and Security Informatics*. IEEE. 2013, pages 129–133.

[6] N. Lohmann. *JSON for Modern C++*. 2016. URL: https://github.com/nlohmann/json (last accessed 2020-04-01).

[7] K. Maeda. "Performance evaluation of object serialization libraries in XML, JSON and binary formats". In: *2012 Second International Conference on Digital Information and Communication Technology and it's Applications* (*DICTAP*). IEEE. 2012, pages 177–182.

[8] C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. "Polonium: Tera-scale graph mining for malware detection". In: *KDD-LDMTA'10*. 2010.

[9] P. Najafi, A. Sapegin, F. Cheng, and C. Meinel. "Guilt-by-Association: Detecting Malicious Entities via Graph Mining". In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2017, pages 88–107.

[10] H. S. Njemanze and P. S. Kothari. *Real time monitoring and analysis of events from multiple network security devices*. US Patent 7,376,969. May 20, 2008.

[11] A. Sapegin, D. Jaeger, A. Azodi, M. Gawron, F. Cheng, and C. Meinel. "Hierarchical object log format for normalisation of security events". In: *2013 9th International Conference on Information Assurance and Security* (*IAS*). IEEE. 2013, pages 25–30.

[12] J. Shenk. "Ninth log management survey report". In: *SANS Institute InfoSec Reading Room* (2014).

[13] The Apache TinkerPop Authors. *GraphSON format description*. 2013.

[14] M. Yip. *Native JSON Benchmark, 2015*. 2015. URL: https://github.com/miloyip/nativejson-benchmark (last accessed 2020-04-01).

Visual Summary of a Telecom Operator's Customer Base

Julia Sidorova and Lars Lundberg

Department of Computer Science
Blekinge Institute of Technology
{ysi,llu}@bth.se

As pointed out by Zadeh, the mission of fuzzy logic in the era of big data is to create a relevant summary of huge amounts of data and facilitate decision making. In this study, elements of fuzzy set theory are used to create a visual summary of telecom data, which gives a comprehensive idea concerning the desirability of boosting an operator's presence in different neighborhoods and regions. The data used for validation cover historical mobility in a region of Sweden during a week. Fuzzy logic allows us to model inherently relative characteristics, such as "a tall man" or "a beautiful woman", and importantly it also defines "anchors", the situations (characterized with the value of the membership function for the characteristic) under which the relative notion receives a unique crisp interpretation. We propose color coding of the membership value for the relative notions such as "the desirability of boosting operator's presence in the neighborhood" and "how well the operator is doing in the region". The corresponding regions on the map (e.g., postcode zones or larger groupings) are colored in different shades passing from green (1) though yellow (0.5) to red (0). The color hues pass a clear intuitive message making the summary easy to grasp.

1 Introduction

Currently, a technology is wanted that is capable of creating a useful summary of multitudes of data. Such a summary was desired to be made in a natural language, but at the same time with a mathematical precision preserved.

Contributions

In our previous research [2, 3, 4, 5] we proposed two types of queries on geodemographic data: 1) the desirability of different geodemographic segments, and 2) operator's current success compared to the best theoretically possible. In both cases the queries return the value of the membership function that is further interpreted. Both types of queries rely on the outcome of the resource allocation module, which operated in the following manner. The problem is that of finding an optimal combination of user segments, given that we want to maximize the overall number of users, who consume finite resources. This problem belongs to a classical family of resource allocation problems for which solutions are found with linear program-

ming (LP) [1]. The LP is defined by the decision variables, the objective function and the restrictions. The idea has been to complement such data summaries with visual summaries (especially that a map is better seen than read about) with intuitively clear color symbolism.

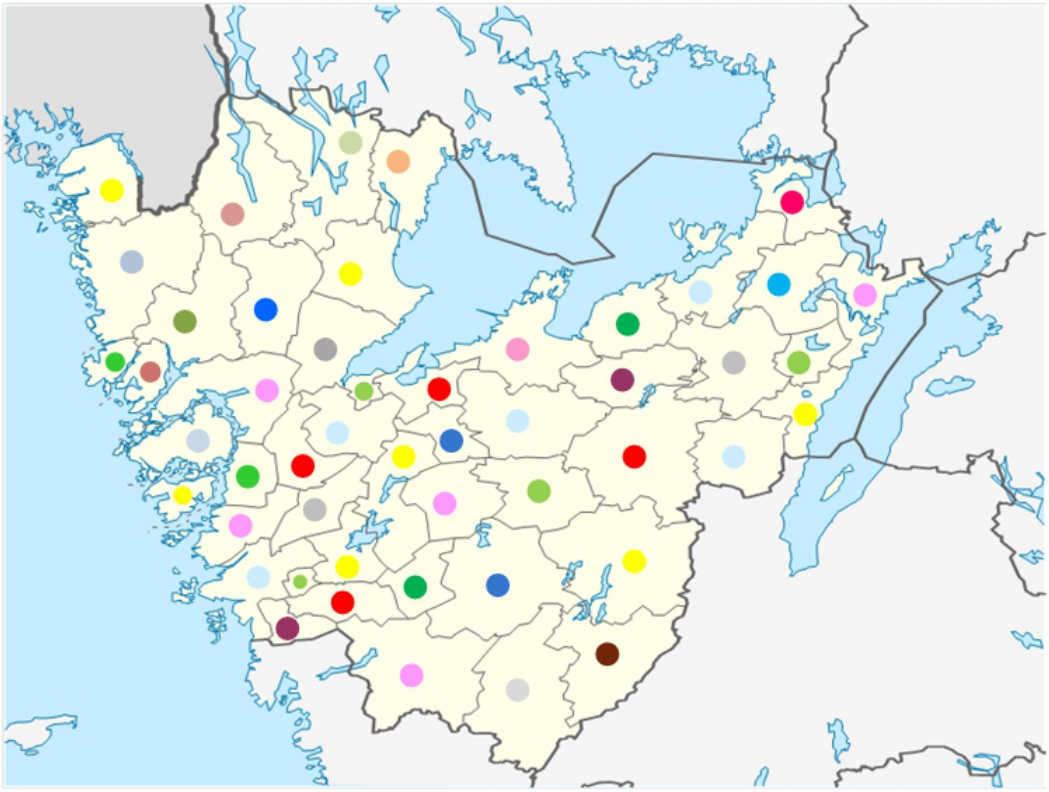


Figure 1: A graphical summary of the telecom operator's presense in the region

2 Geospatial and Geodemographic Data

The study has been conducted on anonymized geospatial and geodemographic data provided by a Scandinavian telecommunication operator. The data consist of CDRs (Call Detail Records) containing historical location data and calls made during one week in a midsized region in Sweden with more than 1000 radio cells. Several cells can be located on the same antenna. The cell density varies in different areas and is higher in city centers, compared to rural areas. The locations of 27010 clients are registered together with the identification of the cells that served them. The client's location is registered every 5 minutes. In the periods when a client does not generate any traffic, she does not make any impact on the infrastructure and such periods of inactivity are not relevant in the light of the resource allocation analysis. These

periods are not used in the present study. Every client in the database is labeled with her geodemographic segment. The fields of the database used in this study are: the cells IDs with the information about the users they served at different time instants, the location coordinates of the cells, the time stamp of every event that generated traffic, and the ID of the user that originated the event, and the MOSAIC geodemographic segment for each client. There are 14 MOSAIC segments present in the database.

3 Results and Conclusions

Instead of tables with fractions in the range of [0,1] representing the scoring of "success regarding infrastructure exploitation" or "desirability of boosting presence in different neighborhoods", the analyst is presented with a colored map. Transmitting information about geographic zones is much handier with colored maps compared to tables <postcode, fraction> or verbal descriptions. The use of such a representation can vary. In Fig. 1, the color of the round tag on the geographic zone symbolizes the operator's marketing success. The table with the mapping of colour to the value in the interval [0,1] is currently done relying on hand-crafted heuristics, and we are working to define such a generic and justifiable association.

References

[1] G. B. Dantzig and M. N. Thapa. *Linear programming 1: introduction*. Springer Science & Business Media, 2006.

[2] S. Podapati, L. Lundberg, L. Skold, O. Rosander, and J. Sidorova. "Fuzzy recommendations in marketing campaigns". In: *European Conference on Advances in Databases and Information Systems*. Springer. 2017, pages 246–256.

[3] J. Sidorova, H. Grahn, O. Rosander, L. Skold, L. Lundberg, and G. Tsihrintzis. "Finding a healthy equilibrium of geodemographic segments for a telecom business". In: *Machine Learning Paradigms-Advances in Data Analytics*. Springer.

[4] J. Sidorova, L. Lundberg, O. Rosander, H. Grahn, and L. Skold. "Recommendations for marketing campaigns in telecommunication business based on the footprint analysis: Who is a good client?" In: *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE. 2017, pages 1–6.

[5] J. Sidorova, L. Sköld, O. Rosander, and L. Lundberg. "Optimizing utilization in cellular radio networks using mobility data". In: *Optimization and Engineering* (2018), pages 1–28.

Improving Test Suite Generation by Testing Google Play's Top 1000 Apps II

Thomas Vogel[1], Chinh Tran[2], Irene Moser[3], and Lars Grunske[1]

[1] Software Engineering Group
Humboldt-Universität zu Berlin, Germany
thomas.vogel@informatik.hu-berlin.de
grunske@informatik.hu-berlin.de
[2] Technische Universität Berlin, Germany
chinh.tran@mail.de
[3] Swinburne University of Technology, Australia
imoser@swin.edu.au

Software testing plays an important role in providing evidence for the correctness of software systems. Due to the costs and complexity of testing such systems, tests are often automatically generated using search-based techniques. The goal of this project is to improve the generation of test suites for mobile applications by adapting existing search techniques based on results from fitness landscape analyses. In this report, we discuss the research context of this project and initial experiments we are conducting in the Future SOC Lab.

1 Introduction

To provide evidence for the correctness of software systems, testing plays an important role, in practice [3, 7] as well as in software engineering research where it is an active topic of research. In the context of search-based software engineering [5], researchers investigate the automated, search-based generation of test data and test suites because manual testing of software systems is complex, costly, and time-consuming [4, 8].

One application area of automated, search-based generation of test suites are mobile applications (apps) [2, 7]. In this area, one approach is *Sapienz* that has discovered bugs (in terms of crashes) in roughly every third Android app that has been tested [7]. This illustrates the relevance of testing and the need to continuously improve test generation approaches.

2 Research Context

In search-based testing, tests are automatically generated by heuristics that – guided by a fitness function – search for optimal tests. The heuristic or generally search technique that is used in an approach is often selected and configured in black-box fashion because the characteristics of the search problem are unknown. Consequently, the

expected performance of a search technique is also unknown. In practice, two possible options of selecting and configuring a search technique are typically used: (1) try out different techniques to identify which one works best for the given problem, or (2) select a popular technique and use its default configuration. Option (1) has the drawback of being costly and time-consuming, and option (2) does not guarantee that the selected technique works well on different specific problems such as different apps (cf. no-free-lunch theorems for search [11, 12]). This requires selecting and configuring search techniques based on the specific search problem, among others, given by the app under test.

For this purpose, we aim for a better understanding of the search problem by analyzing the fitness landscape [6, 9]. These analysis results will provide feedback for selecting and especially configuring the search technique for a test generation approach so that better tests are generated, for instance, with respect to achieved coverage and number of revealed faults.

Particularly, we consider the application area of generating test suites for Android apps and aim for improving the state-of-the-art approach Sapienz [7]. Therefore, we plan a large study of testing the top 1000 apps of the Google Play store. This study will further provide a report about the state of the practice in how reliable today's most popular apps are.

The interested reader is referred to [10] for a refined discussion of the research problem and goals of this project.

3 Initial Experiments and Preliminary Results

Based on a preliminary fitness landscape analysis of generating test suites for apps, we adapted the search technique of Sapienz. These adaptations aim at preserving the diversity of test suites during the search in order to eventually generate better tests. To evaluate these adaptations, we are currently conducting an initial experiment using the 68 app benchmark collected by Choudhary et al. [2]. In this experiment, we perform a head-to-head comparison between the original and the adapted Sapienz.

The use of this benchmark is motivated by two aspects. First, the apps of the benchmark are open source so that we are able to measure the code coverage achieved by the generated tests. This coverage metric is more fine-grained and precise than the activity coverage to assess the effectiveness of different test generation approaches. Second, this benchmark can be seen as an initial test of our work before scaling up to 1000 apps.

Preliminary results indicate that our adapted version of Sapienz is able to reveal more faults than the original Sapienz. Given a time budget of one hour to generate tests for each app, the adapted Sapienz revealed in total 166 faults while the original Sapienz revealed 140 faults for all 68 apps. These preliminary results are promising and need to be refined considering the other objectives (fitness dimensions) of test suites, which are length of test sequences and achieved (code) coverage.

4 Used HPI Future SOC Lab Resources

The experiments were performed on eight blade servers (*hum nodes*) of the HPI Future SOC Lab. To parallelize the execution, at any point in time during an experiment one app was tested on each server. For each server, the evaluations of test suites (i.e., the execution of multiple test suites) for one app was parallelized by running ten Android emulators concurrently (cf. deployment and distribution model [10]).

5 Conclusion and Next Steps

In this report, we have discussed the research context (cf. [10]) and our initial experiments. As the next steps, we will extend the experiments to enable an inferential statistical analysis of the results. This requires repetitions (around 30 [1]) of the experiments to obtain a sample of runs for the statistical analysis. The analysis will provide solid evidence for the improvements of test suite generation for apps achieved by our work. Moreover, we will run the large-scale study in the Future SOC Lab that scales the experiment to Google Play's top 1000 apps.

References

[1] A. Arcuri and L. Briand. "A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering". In: *Software Testing, Verification and Reliability* 24.3 (2014), pages 219–250. DOI: 10.1002/stvr.1486.

[2] S. R. Choudhary, A. Gorla, and A. Orso. "Automated Test Input Generation for Android: Are We There Yet? (E)". In: *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. ASE '15. Washington, DC, USA: IEEE Computer Society, 2015, pages 429–440. ISBN: 978-1-5090-0025-8. DOI: 10.1109/ASE.2015.89.

[3] G. Fraser and A. Arcuri. "1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite". In: *Empirical Software Engineering* 20.3 (2015), pages 611–639. DOI: 10.1007/s10664-013-9288-2.

[4] M. Harman. "The Current State and Future of Search Based Software Engineering". In: *Workshop on the Future of Software Engineering, FOSE*. 2007, pages 342–357. DOI: 10.1109/FOSE.2007.29.

[5] M. Harman, S. A. Mansouri, and Y. Zhang. "Search-based Software Engineering: Trends, Techniques and Applications". In: *ACM Comput. Surv.* 45.1 (2012), 11:1–11:61. DOI: 10.1145/2379776.2379787.

[6] K. M. Malan and A. P. Engelbrecht. "A survey of techniques for characterising fitness landscapes and some possible ways forward". In: *Information Sciences* 241 (2013), pages 148–163. DOI: 10.1016/j.ins.2013.04.015.

[7] K. Mao, M. Harman, and Y. Jia. "Sapienz: Multi-objective Automated Testing for Android Applications". In: *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ISSTA 2016. ACM, 2016, pages 94–105. DOI: 10.1145/2931037.2931054.

[8] P. McMinn. "Search-based Software Test Data Generation: A Survey". In: *Software Testing, Verification and Reliability* 14.2 (2004), pages 105–156. DOI: 10.1002/stvr.v14:2.

[9] E. Pitzer and M. Affenzeller. "A Comprehensive Survey on Fitness Landscape Analysis". In: *Recent Advances in Intelligent Engineering Systems*. Springer, 2012, pages 161–191. DOI: 10.1007/978-3-642-23229-9_8.

[10] T. Vogel, C. Tran, I. Moser, and L. Grunske. *Improving Test Suite Generation by Testing Google Play's Top 1000 Apps*. Technical report. HPI Future SOC Lab Project Reports, 2018.

[11] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pages 67–82. DOI: 10.1109/4235.585893.

[12] D. H. Wolpert and W. G. Macready. *No Free Lunch Theorems for Search*. Technical report SFI-TR-95-02-010. Santa Fe Institute, 1995.

Machine Learning Approach for Live Migration Cost Prediction in VMware Environments

Mohamed Esam Elsaid[1], Hazem M. Abbas[2], and Christoph Meinel[1]

[1] Hasso Plattner Institute, Potsdam, Germany
{Mohamed.Elsaid,Meinel}@hpi.de
[2] Dept. Computer and Systems Engineering
Ain Shams University, Cairo, Egypt
Hazem.Abbas@eng.asu.edu.eg

Virtualization became a commonly used technology in datacenters during the last decade. Live migration is an essential feature in most of the clusters hypervisors. Live migration process has a cost that includes the migration time, downtime, IP network overhead, CPU overhead and power consumption. This migration cost cannot be ignored, however datacenter admins do live migration without expectations about the resultant cost. Several research papers have discussed this problem, however they could not provide a practical model that can be easily implemented for cost prediction in VMware environments. In this paper, we propose a machine learning approach for live migration cost prediction in VMware environments. The proposed approach is implemented as a VMware PowerCLI script that can be easily implemented and run in any vCenter Server Cluster to do data collection of previous migrations statistics, train the machine learning models and then predict live migration cost. Testing results show how the proposed framework can predict live migration time, network throughput and power consumption cost with accurate results and for different kinds of workloads. This helps datacenters admins to have better planning for their VMware environments live migrations.

1 Introduction

Datacenter resource virtualization is commonly used by IT administrators during the last decade. Running virtual or software defined machines has shown higher availability, rapid scaling, better resource utilization and more cost efficiency. Live migration of virtual machines is a key feature in virtual environments and cloud computing datacenters. Using live migration, virtual machines can be moved from a physical host to another while the applications are running online. This is because live migration causes negligible service interruption during the migration process. Servers load balance, power saving, fault tolerance and dynamic virtual machines allocation are all dependent on live migration [1]. During the live migration process, the VM CPU cache, memory pages and IO buffers contents are migrated. However the storage content is shared between the source and the target servers, so storage

content is not migrated. Live migration is supported by VMware (vMotion), Xen (XenMotion), Microsoft Hyper-V and KVM.

For the best of our knowledge, until now live migration is done by datacenter admins with no expectations about the migration cost. So the admins do the live migrations and then see the impact of it on the network, CPU, memory and power consumption. This leads sometimes to facing failures in live migration, bottlenecks in the datacenter infrastructure resources, and downgrade in the VM services availability due to longer down time. These problems happen especially for large memory VMs migrations.

In this paper, we propose a practical machine learning framework for live migration cost modelling and prediction in VMware environments. The proposed framework starts with data collection about the history of live migrations that have run in the cluster during the past 12 hours, then use the collected statistics for models training. After training the models, the prediction phase can start to estimate the future live migration requests cost. This should help the datacenters admins to estimate the cost of single or multiple VMs live migration before proceeding with it. This means better live migration tasks planning, less resources bottlenecks and so higher service availability during live migration. Machine learning is a subset of Artificial Intelligence (AI) which is now used in many applications that touch our daily life activities including health care, retail, social networks, aerospace, autonomous driving and IT industries. We focus in this research on using machine learning for VMware vMotion cost prediction, as one of the widely used hypervisors in enterprise level datacenters. The obtained empirical models are based on using a VMware cluster test-bed.

2 Proposed Cost Prediction Framework

In this paper, we solve the challenge of having a practical live migration cost prediction for VMware environments. This is achieved by proposing a machine learning based approach that is implemented as VMware PowerCLI script [6] and can connect to any VMware vCenter server [5] to train the model and then predict live migration cost. This prediction is showed as alert to the cluster admin to know the expected cost of live migration before proceeding with it. We list our contribution in this paper in the following points:

- We propose a machine learning approach for VMware vMotion that predicts the live migration time, network overhead and power consumption given the active memory size of the VM.
- The proposed approach can be practically used. It is implemented as a VMware PowerCLI script [6]; that can be bounded with any VMware vCenter server [5] and show the cost prediction results to the cluster admin whenever live migration is requested as an important note before proceeding with live migration. This helps the network admins to avoid resources bottlenecks and minimize live migration failures due to resources contention.

- The proposed script includes data collection that is used for the training phase. This makes the proposed models adaptable to each VMware cluster automatically. The collected data are the performance statistics of the past 12 hours; specifically the VM active memory size, the migration time, the transmission rate and the peak power overhead.
- The training phase in the proposed machine learning algorithm is fed by the live migration operations that run in the datacenter; which makes that the prediction accuracy increases with more live migrations happen in the cluster.

2.1 Modelling of Live Migration Cost

This paper is an extension to the proposed models in [2] and [3]. From these papers, the following empirical models could be proposed for live migration time, data rate and power consumption after applying the regression techniques.

- The relation between the network rate and the active memory size can be modelled as an exponential relation; as shown in equation (1).

$$R_s = \alpha e^{V_{Mem}} + \beta \tag{1}$$

  R_s is the source host network throughput overhead in kBps, V_{mem} is the source host active memory size in kB at the time when the live migration should start. α and β are the equation constants. From equation (1), we note that even if the memory content is zero, the transmission rate will have a value. This value represents the CPU and buffers content that should be migrated.

- Migration Time: A linear relationship is obtained between the migration time and the division of the memory size over the transmission rate; as represented in equation (2).

$$T_{mig} = a.(\frac{V_{mem}}{R_s}) + b \tag{2}$$

  T_{mig} is the migration time duration in seconds. a and b are the equation constants. From equation (2), we note that even is there is no active memory content, the migration time will be the value b; which represents the CPU and adapters content to be migrated.

- Peak power consumption overhead has linear relation with the transmission rate; as represented in equation (3).

$$P_{mig} = \frac{dE_{mig}}{dt} = c\ \frac{dV_{mig}}{dt} = c\ R_s \tag{3}$$

  P_{mig} is the peak power overhead in Watt, and c is constant.From equation (3), we note that, if there is no change in the datarate; which means that there was no data to send from the source host, the peak power change will be zero.

In our previous papers [2] and [3], the above models could be used for cost analysis but not for cost prediction. This is because of the equations constants. These constants depend on the cluster hardware configuration like CPU specs, so they change from a cluster environment to another. So in order to determine these constants and achieve higher accuracy in cost prediction, we propose a machine learning framework that helps in obtaining these constants for each cluster and so be able to predict the live migration cost. This is what we discuss in more details in the next section.

2.2 Machine Learning based Cost Prediction

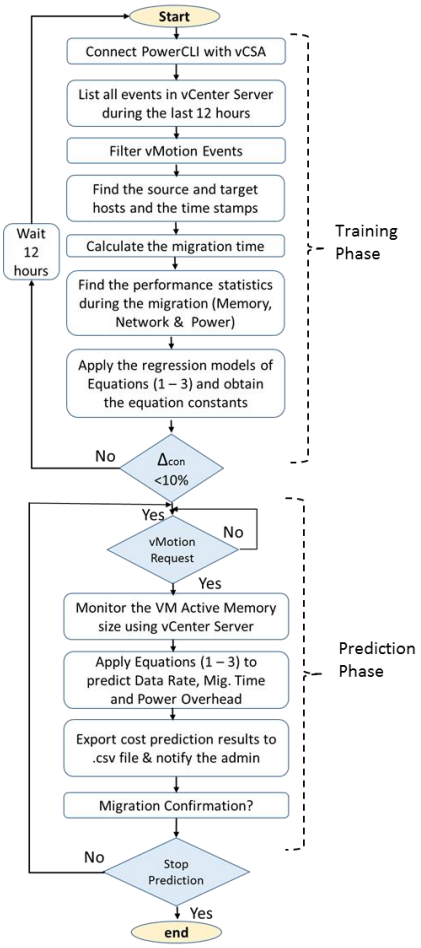


Figure 1: Proposed Prediction Framework

In this paper, machine learning is used because the proposed models in equations (1–3) can not be used in live migration cost prediction. This is due to the constants included in the equations. These constants values depend on the cluster hardware characteristics; like CPU and network configurations. So, machine learning is needed

to train the models in reference to equations (1–3) until the constants values are obtained for each cluster. Then, these equations can be used for cost prediction.

In this section, we present the proposed machine learning based framework for live migration time, transfer rate and power consumption overhead prediction. As shown in the flow chart of figure 1, the proposed framework consists of two main phases, the training phase and the prediction phase.

The training phase starts when the VMware PowerCLI script connects to the cluster vCenter Server Appliance (vCSA). Then data collection starts with listing all the events happened in the cluster during the last 12 hours. This 12 hours cycle can be changed based on the cluster admin preference. From the collected events, live migration; vMotion events are filtered. These vMotion events details like the source host, target host and time stamp are captured. The time stamp include the start time and the complete time of the vMotion event. Then the script calculates the complete and start time differences in order to get the migration time of each vMotion request. The performance logs of vCSA are collected at the start and the completion times at the vMotion events in order to get the active memory size of the migrated VMs in kB, the network overhead in kBps and the peak power change in Watt.

From the above data of each vMotion event, we use the regression models in equations (1–3) to calculate the equations constants after doing several substitution and considering the minimum Root Mean Square Error (RMSE); equation (4).

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(d_i - f_i)^2} \tag{4}$$

Where N is the number of sample points collected during the last 12 hours. d_i is the measured performance value and f_i is the regression model equation value.

If the change in all the constants value became greater than 10 % of the last 12 hours cycle, the script waits for more 12 hours and run again to continue in the training phase. If these changes became less than 10 % of the last 12 hours, so we consider the the training phase of this cluster is finished, and the script then moves to the prediction phase. The time consumed until reaching this 10 % convergence depends on the changes happen mainly in the VMs active memory size; which depends on the running workloads. This means that the more changes happen in the active memory size of the VMs, the less time required to reach the 10 % convergence; this is because the regression relation curve will cover most of the values that the active memory size can take and so the regression relation model can be defined faster.

This sequence of data collection and models training makes the algorithm can fit at any vCenter Server cluster and adapt its models based on the cluster configuration in order to provide cost prediction with high accuracy.

In the prediction phase when a vMotion request is sent by the cluster admin, the active memory size is captured by the script before proceeding with live migration. Once the active memory size is known, equation (1) is used to predict the source host network throughput. Then equation (2) is used to predict the migration time, and finally equation (3) is used to predict the peak power consumption. The prediction data is exported to a .csv file that the cluster admin can read, and then decide either to proceed with this migration or not.

3 Testing Environment

The testing environment is shown in figure 2; as shown it has a similar infrastructure to enterprise datacenters. It includes the following hardware setup; three Hosts (Hewlett Packard DL980 G7) with 8 x Intel Xeon (Nehalem EX) X7560, 8 GB RAM, 4 NICs, 2 HBA with 2 Fiber ports per card. The three hosts are connected to a shared storage EMC VNX5800; 1TB LUN via FC-SAN network.The Ethernet switch is Cisco with 1Gbps ports. From software prespective, VMware ESXi 6.5.0 Hypervisor is used with vCenter Server that manages both hosts and the VMs live migration. VMware PowerCLI 6.5.1 build 5377412 is connected to the vCenter Server to run the framework algorithm script.

In this set up we have created four Linux Ubuntu 12.04 VMs with 4 vCPU, and different RAM sizes (1 GB, 2 GB, 4 GB and 8 GB). The VMs have mainly 3 categories of workload:

- CPU and Memory intensive: This is considered as the worst case scenario for a running workload. The CPU intensive benchmark that we used is Linpack [4] and the memory stress is the Linux *Stress* Package [8].
- Network Intensive: The network stress benchmark that we have used is Apache Bench (AB). Apache Bench tool stresses the web servers with lots of requests through the network to test the servers response.
- Idle: VM is simply an idle Ubuntu OS VM; with no running applications.

From networking side, we have followed VMware best practice to isolate live migration traffic by creating a dedicated VMkernel vMotion port group [7]. As a basic introduction to networking in VMware vCenter server, a virtual Distributed Switch (vDS) is created between the cluster physical hosts through the hosts physical network adapters. This vDS is connecting all the VMs insides these physical hosts; such that each VM is connected to this vDS using its virtual ports. A port group is a aggregation of multiple virtual ports to isolate their traffic using labeling. This is used basically to isolate management, kernel and data traffic from each other. vMotion has a specific port group in order to isolate its traffic; as a kind of VMkernel traffic [7]. With this testing setup, we have run 12 testing scenarios; as a matrix of 3 workload categories and 4 different VM sizes. For each configuration, we have run live migration at least 10 times. So the resultant is 144 readings. For each run, we do live migration for a VM from one of the physical hosts to other of the other two hosts; without doing any storage migration [9]. So only the CPU state, memory and buffers contents are migrated. For data collection and models training, every 12 hours the script gathers all the live migration events that happened in the cluster and from the events timing details, the live migration can be obtained. Then the script uses $Get - Stat$ function to import the statistics beyond each live migration event; specifically the data rate change, the peak power increase and the active memory size of the migrated VMs. This data is used for models training for the cluster.

For the prediction phase, we make use of the trained models to predict the future VM live migration cost when the admin sends a live migration request, and given

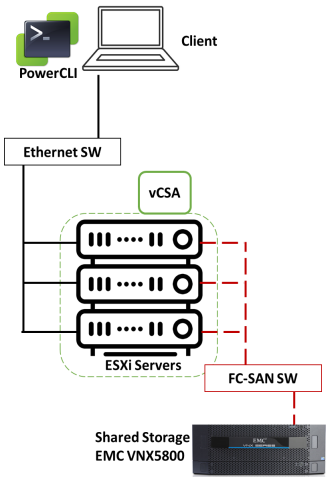


Figure 2: Testing Lab Layout

the active memory in kB. The estimated cost is exported as csv file that the admin can check before proceeding with the live migration.

4 Results and Analysis

After testing the proposed approach in figure 1 on the test-bed of figure 2, we present in this section the prediction results for almost 144 readings. Before showing the prediction phase graphs, we start with the training phase. This is to show how the models are trained until obtaining equations (1–3) constants with at least 90 percent accuracy.

4.1 Training Phase

In this phase, the script collects the last 12 hours live migration events. Then the performance statistics of these live migrations are gathered; give the time stamps of the events. The migration time is calculated by the script; given the start and end time of the live migration event. The other gathered statistics include the active memory size, the source host transmission rate and the peak power change. All these details are used to train the models of equations (1–3) and to obtain the constants of this cluster by solving several linear equations. For example in order to calculate a and b of equation (2), we use every two live migration events statistics to generate two equations in two unknowns. These unknowns are a and b in this example, because the migration time, the active memory size and the transmission rate are given. So, we gather every two live migration events statistics to solve for the constants of equation (2). The script keeps on solving for the values of a and b until finding the changes in the values of a and b are less than 10 percent compared to the calculated values

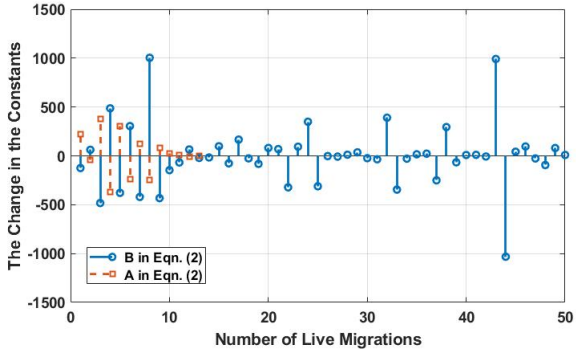


Figure 3: A and B Change until Saturation

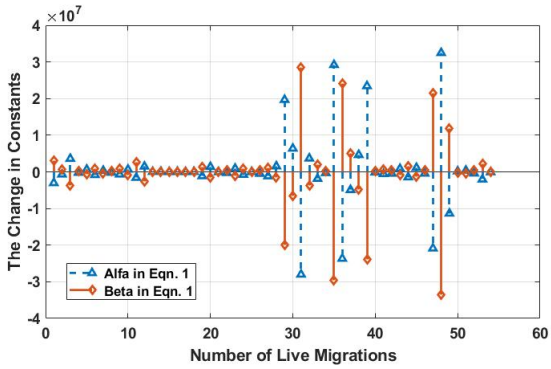


Figure 4: Alpha and Beta Change until Saturation

of last equations solution. This means that these constants are at least 90 percent saturated. Figure 3 shows the changes of a and b constants versus the number of live migration equations that were used until reaching the 90 percent saturation. As shown in figure 3; the difference in a is changing with the number of live migrations which represents solving more equations until the 90 percent saturation at difference equals 0.22 after 14 live migrations. At this point a=9.04. For b constant, the script has run 50 live migrations to reach the 90 percent saturation at difference equals 9.16. At this point b=21.04. This means that modeling with equation (2) could be used after 50 live migration runs for this cluster. For equation (1), we could also solve every two equations of live migrations data as linearly to obtain the values of α and β. The is because the values of the active memory size and the migration time are given, so we can substitute with them and then solve two equations in two unknowns; α and β. Figure 4 shows the differences happen in the values of α and β after each live migration until reaching the 90 percent saturation. As shown; the constant α could reach the saturation at difference equals 1850 after 54 live migrations. At this point α equals $2.02 * 10^4$. The value of β reaches the 90 percent saturation at difference equals 2225 also after 54 live migrations. At this point, β equals $2.33 * 10^4$. This means that modeling with equation (1) can be used after 54 live migration runs. Finally, equation (3), which has just one unknown; c and so it can be resolved given just

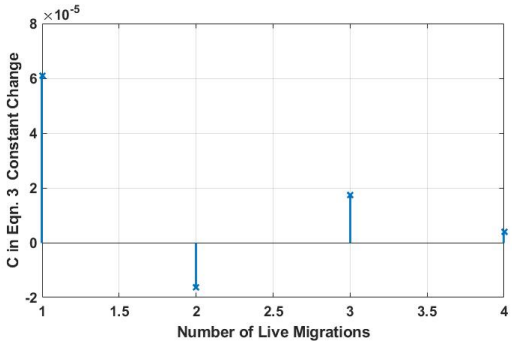


Figure 5: C Change until Saturation

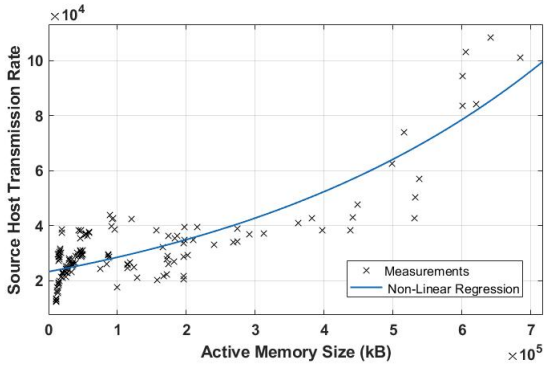


Figure 6: Rate vs Active Memory Size

one live migration statistics. So for each live migration run in the past 12 hours, we could read the transmission rate and the peak power overhead and then calculate the constant c. Figure 5 shows the changes happen with each live migration calculation to the the constant c; as shown it 90 percent saturates after just fours live migrations runs at difference of c value equals $0.6 * 10^{-5}$. At this point c equals $16 * 10^{-5}$.

From the above analysis, we find that it required 54 live migration runs to be able to train the models provided in equation (1–3). In general, the required number of live migrations runs to finish the training phase depends on the error gap between the training data and the regression model. The closer gap between the training data set and the model, the lower number of live migration iterations required to reach the 90 percent saturation, and vice verse.

4.2 Prediction Phase

In this subsection, we build on the training phase that we have discussed above. Now, the regression models are trained for this cluster and ready to be used for future live migration cost prediction. The testing results in figures 6–8, show the regression models that are used and the actual measured data after migration. The measurement points in figure 6, figure 7 and figure 8 are VMs live migrations with different

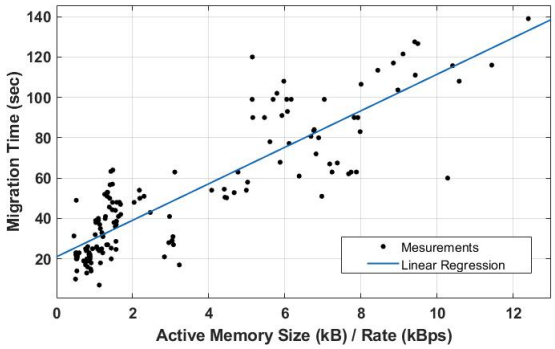


Figure 7: Migration Time vs Active Memory Size/Rate

configurations including memory size of 1 GB, 2 GB, 4 GB and 8 GB VMs that utilize three different kinds of workloads. As discussed in section V, these workloads are CPU and memory intensive, network intensive and idle VMs. This results in 12 different VM configurations. Each configuration is tested 12 times; which represents the existing 144 measurement points in the following figure. The prediction starts with figure 6; so given the active memory size of the VM to be migrated, the source host transmission rate can be predicted. The VM active memory size can be measured before live migration. Figure 6 shows the exponential relation as a valid regression model between the active memory size and the transmission rate. That is how the transmission rate can be predicted. Table 1 shows the RMSE of figure 6 in reference to equation (4). After obtaining the transmission rate from figure 6, we calculate now the active memory size over the transmission rate; which is the horizontal axis of figure 7. So the migration time can be predicted; using the linear regression model of figure 7. The RMSE of the prediction in figure 7 is also listed in table 1. Figure 7 also shows that the migration time can consume several minutes in case of large memory and memory intensive VMs. The last model is for the source host peak power change; which is shown in figure 8. So given the source host transmission rate, the peak power change can be obtained using linear regression. All these predicted

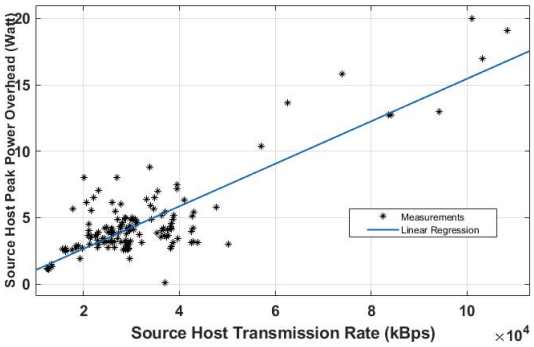


Figure 8: Peak Power Overhead vs Transmission Rate

Table 1: RMSE of the Regression Models

Model	Fig.	RMSE
Transmission Rate	figure 6	8187
Migration Time	figure 7	15.5
Peak Power	figure 8	1.7

live migration cost parameters are exported to a .csv file that can be accessed by the cluster admin to check the estimated cost if he/she decides to do live migration to a certain VM. This help the admins to have better planning for live migrations, and avoid resource bottlenecks that lead to live migration failures and service quality degradation. This proposed framework script can adapt itself by changing the models constants using the training phase; which make it flexible with any VMware cluster.

5 Conclusion

Live migration is an essential features in modern datacenters and cloud computing environments. Migration time, network throughput and power consumption overhead are part of the live migration cost. This cost can not be ignored and might lead to resources bottlenecks, service availability degradation and live migration failures. Several related papers have discussed this problem by applying mathematical and empirical studies, however to the best of our knowledge there is no related paper that could provide a practical approach that can be used and integrated with VMware clusters. In this paper, we proposed a practical machine learning based approach that helps the datacneter admins to predict the live migration cost in VMware environments. The proposed framework is implemented as VMware PowerCLI script and can connect to any vSphere vCenter Server. We considered simplicity in the proposed approach to minimize the CPU consumption overhead due to running the proposed approach and so make it agile enough to be implemented in enterprise datacenters. The algorithm starts with data collection for the live migration history in the past 12 hours, and then use this data for models training. Then the prediction phase starts by using the VM active memory size as a given, and the regression model to predict the live migration cost. In this paper, as could predict the live migration time, network throughput and power consumption overhead. Testing results section shows the sequence that the script follows in the training phase. It shows also that the proposed regression based models can be used for cost prediction with acceptable error. In the future work, we will consider adding real time model training that can work in parallel with cost predictions. Also we will consider multiple VMs migrations cost prediction testing.

References

[1] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil. "A critical survey of live virtual machine migration techniques". In: *J. Cloud Comput.* 6.1 (Dec. 2017).

[2] M. E. Elsaid and C. Meinel. "Live migration impact on virtual datacenter performance: Vmware vmotion based study". In: *2014 International Conference on Future Internet of Things and Cloud*. IEEE. 2014, pages 216–221.

[3] M. E. Elsaid and C. Meinel. "Multiple Virtual Machines Live Migration Performance Modelling - VMware vMotion Based Study". In: *2016 IEEE International Conference on Cloud Engineering, IC2E 2016, Berlin, Germany, April 4-8, 2016*. 2016, pages 212–213. DOI: 10.1109/IC2E.2016.9.

[4] *Linpack*. URL: https://www.netlib.org/linpack/ (last accessed 2019-02-28).

[5] *vCenter Server*. URL: https://www.vmware.com/products/vcenter-server.html (last accessed 2019-02-28).

[6] *VMware PowerCLI*. URL: https://code.vmware.com/tool/vmware-powercli/6.5 (last accessed 2019-02-28).

[7] *vSphere ESXi Server Networking Guide*. URL: https://docs.vmware.com/en/VMware-vSphere/5.5/vsphere-esxi-vcenter-server-552-networking-guide.pdf (last accessed 2019-02-28).

[8] A. Waterland. *stress*. July 18, 2014. URL: https://web.archive.org/web/20190702093856/http://people.seas.harvard.edu/~apw/stress (last accessed 2019-07-02).

[9] J. Zheng, T. S. E. Ng, and K. Sripanidkulchai. "Workload-aware live storage migration for clouds". In: *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '11)*. 2011, pages 133–144.

Quality Assessment of RDF datasets at Large Scale

Anisa Rula[1], Gezim Sejdiu[2], and Andrea Maurino[1]

[1] University of Milano-Bicocca
{firstname.lastname}@unimib.it
[2] University of Bonn
{lastname}@cs.uni-bonn.de

Over the last years, Linked Data has grown steadily. Today, we count more than 10,000 datasets being available online following linked data standards. Thanks to these standards, data in machine-readable and interoperable formats is available. Nevertheless, many applications, such as data integration, search, and interlinking, cannot take full advantage of Linked Data if it is of low quality. In fact, there are already a few approaches, which offer the quality assessment of Linked Data. However, those works showed deficiencies in terms of performance once the dataset size grows beyond the capabilities of a single machine. In this report, we proposed the execution of quality assessment of large RDF datasets that can scale out to a cluster of machines. We describe the first distributed in-memory approach for computing different quality metrics for RDF datasets using Apache Spark.

1 Introduction

Creating and managing large-scale RDF datasets has been the key to success for many applications, such as semantic search, query answering and machine reading [8]. The quality of such knowledge bases plays a fundamental role in large-scale data consuming applications. Traditional techniques of quality assessment for RDF datasets are not adequate to assess the quality at large-scale. These approaches mostly fail to capture the new dimensions (the four Vs) of big data.

A limited number of solutions have been conceived to offer quality assessment of Linked Data [1, 2, 3, 5]. However, these methods can be mainly used on a small portion of large datasets [5] or narrow down to specific problems e.g., syntactic accuracy of literal values [1] or accessibility of resources [7]. In general, these existing efforts show severe deficiencies in terms of performance when the data grows beyond the capabilities of a single machine. This limits the applicability of existing solutions to medium-sized datasets only, in turn, paralyzing the role of applications in embracing the increasing volumes of the available datasets.

We are working on a novel approach for computing RDF dataset quality metrics and implement it using an efficient framework for large-scale, distributed and in-memory computations. Within this project, we will focus on performing analysis of the complexity of the computational steps and the data exchange between nodes in

the cluster. We will integrate the approach into the SANSA framework[3] which is a big data processing engine for scalable processing of large-scale RDF data.

The computation of the set of quality metrics in our approach is performed in four steps as follows:

- Defining quality metrics parameters. Definitions are kept on a dedicated file which contains most of the configurations needed for the system to evaluate quality metrics and gather result sets.
- Retrieving the RDF data. RDF data needs first to be loaded into a large-scale storage that Spark can efficiently read from. For this purpose, we use HDFS (Hadoop Distributed File-System). HDFS is able to accommodate any type of data in its raw format, horizontally scale to an arbitrary number of nodes, and replicate data among the cluster nodes for fault tolerance. In such a distributed environment, Spark adopts different data locality strategies to try to perform computations as close to the needed data as possible in HDFS and thus avoid data transfer overhead.
- Parsing and mapping RDF into the main dataset. In the course of Spark execution, data is parsed into triples and loaded into an RDD.
- Quality metric evaluation. For each metric, Spark generates an execution plan, which is composed of one or more of the following Spark transformations: map, filter, reduce and group-by.

We will make use of Apache Spark, using the Scala programming language. The approach that we propose in this project will be similar to the implementation provided in the work DistLODStats: Distributed Computation of RDF Dataset Statistics. In particular, we will perform similarly the first two steps: saving the RDF data in scalable storage and parsing and mapping the RDF data. The difference will be in the last step where instead of profiling statistics about the dataset we will measure quality metrics.

In this report we are able to show the implementation and the first settings. We plan to ask for the resources again in order to run the experiments as planned in the proposal.

2 Implementation

In this section, we give an overall description of the data model and the architecture of our approach. We model and store RDF graphs based on the basic building block of the Spark framework, RDDs. RDDs are in-memory collections of records that can be operated in parallel on a large distributed cluster. RDDs provide an interface based on *coarse-grained* transformations (e.g *map*, *filter* and *reduce*): operations applied on an

[3] http://sansa-stack.net/ (last accessed 2020-04-01).

entire RDD. A *map* function transforms each value from an input RDD into another value while applying the rules. A *filter* transforms an input RDD to an output RDD, which contains only the elements that satisfy a given condition. *Reduce* aggregates the RDD elements using a specific function.

The computation of the set of quality metrics is performed using Spark. We have used the Scala[4] programming language API in Apache Spark to provide the distributed implementation of the proposed approach. Our approach constructs the *main dataset* while reading RDF data (e.g. NTriples file or any other RDF serialization format) and converts it into RDD of triples. This latter undergoes the transformation operation of applying the filtering through rules and producing a new *filtered* RDD. At the end, will serve as an input to the next step which applies a set of actions that are effectively applied. The output of this step will be the metric output represented as a numerical value. The result set of different quality metrics can be further visualized and monitored using SANSA-Notebooks [4].
The user can choose to extract the input in a machine-readable format. We have used the data quality vocabulary[5] (DQV) to represent the quality metrics.

The work done here has been integrated into SANSA [6], an open source[6] *data flow processing engine* for scalable processing of large-scale RDF datasets. SANSA uses Spark and Flink[7] which offer fault-tolerant, highly available and scalable approaches to process massive sized datasets efficiently. SANSA provides the facilities for semantic data representation, querying, inference, and analytics at scale.

3 Evaluation

The major aim of the proposed approach is to serve massive large-scale real-life RDF datasets. We are interested in addressing the following additional questions.

- **Flexibility**: How fast our approach processes different types of metrics?
- **Scalability**: How large are the RDF datasets that our quality assessment approach can scale to? What is the system speedup w.r.t the number of nodes in a cluster mode?
- **Efficiency**: How well our approach performs on real-world datasets?

In the following, we present our experimental setup including the datasets used.

[4] https://www.scala-lang.org/ (last accessed 2020-04-01).
[5] https://www.w3.org/TR/vocab-dqv/ (last accessed 2020-04-01).
[6] https://github.com/SANSA-Stack (last accessed 2020-04-01).
[7] https://flink.apache.org/ (last accessed 2020-04-01).

4 Conclusion

The data quality assessment becomes challenging with the increasing sizes of data. Many existing tools mostly contain a customized data quality functionality to detect and analyze data quality issues within their own domain. However, this process is both data-intensive and computing-intensive and it is a challenge to develop fast and efficient algorithms that can handle large scale RDF datasets.

In this report, we have discussed about our novel approach for distributed in-memory evaluation of RDF quality assessment metrics implemented on top of the Spark framework.

The benefit of using Spark is that its core concepts (RDDs) are designed to scale horizontally. Users can adapt the cluster sizes corresponding to the data sizes, by dropping when it is not needed and adding more when there is a need for it.

References

[1] W. Beek, F. Ilievski, J. Debattista, S. Schlobach, and J. Wielemaker. "Literally better: Analyzing and improving the quality of literals". In: *Semantic Web* 9.1 (2018).

[2] J. Debattista, S. Auer, and C. Lange. "Luzzu—A Methodology and Framework for Linked Data Quality Assessment". In: *Journal of Data and Information Quality (JDIQ)* 8.1 (2016), page 4.

[3] J. Debattista, C. Lange, S. Auer, and D. Cortis. "Evaluating the quality of the LOD cloud: An empirical investigation". In: *Semantic Web* 9.6 (2018), pages 859–901. DOI: 10.3233/SW-180306.

[4] I. Ermilov, J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, S. Bin, N. Chakraborty, H. Petzka, M. Saleem, A.-C. N. Ngonga, and H. Jabeen. "The Tale of Sansa Spark". In: *16th International Semantic Web Conference, Poster & Demos*. 2017.

[5] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger. "Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO". In: *Semantic Web* 9.1 (2018), pages 77–129.

[6] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. Ngonga Ngomo, and H. Jabeen. "Distributed Semantic Analytics using the SANSA Stack". In: *Proceedings of 16th International Semantic Web Conference - Resources Track (ISWC'2017)*. 2017.

[7] N. Mihindukulasooriya, R. García-Castro, and A. Gómez-Pérez. "LD Sniffer: A Quality Assessment Tool for Measuring the Accessibility of Linked Data". In: *Knowledge Engineering and Knowledge Management*. Cham: Springer International Publishing, 2016, pages 149–152. ISBN: 978-3-319-58694-6.

[8] A.-C. Ngonga Ngomo, S. Auer, J. Lehmann, and A. Zaveri. "Introduction to Linked Data and Its Lifecycle on the Web". In: *Reasoning Web*. 2014.

Towards a GPU-Accelerated Skeleton Discovery Beyond Device Memory Capacity

Christopher Schmidt[1], Johannes Hügle[1], and Matthias Uflacker[1]

Enterprise Platform and Integration Concepts
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

Learning the causal structures in high-dimensional datasets provides an opportunity to derive insights from observational data. For example in medicine, gene regulatory networks are a practical embodiment of systems biology and can be applied in diagnostics, or drug design. Algorithms for the derivation of causal relationships, such as the well-known PC algorithm, suffer from long execution times. By harnessing the compute power of Graphics Processing Units (GPUs) the execution time is drastically improved. Yet, current GPU-accelerated implementations are restricted to datasets that entirely fit into the GPU memory.

In this technical report, we address this limitation and propose a block-wise version of the skeleton discovery for execution on the GPU. The approach splits the dataset into smaller data blocks that fit into device memory. Furthermore, it ensures that all possible conditional independence (CI) tests to derive the causal structure can be conducted, even, if CI tests require data from different data blocks. An evaluation of the approach is planned in future work.

1 Introduction and Background

Learning causal relationships between observed variables in complex systems supports human understanding, and allows to gain new insights. For example in genetic research, gene expression data of thousands of genes is a foundation for the construction of gene regulatory networks, used in drug design or diagnostics [7].

Constraint-based algorithms, such as the PC algorithm developed by Spirtes et al. [9], allow for derivation of causal structures from observational data. These algorithms have the goal to find all possible directed acyclic graphs (DAGs) from the observational data. Hence, the causal relationships are depicted in a causal graphical model, in which nodes represent the variables, which are connected by an edge if a causal relationship exists [6]. The causal graphical model is computed through the application of CI tests, appropriate to the underlying distribution of the involved variables [2]. While the first step of the PC algorithm discovers the skeleton graph of the causal model, the second step orients the edges within this skeleton graph to determine the complete partially directed acyclic graph (CPDAG). For details, we refer the reader to Sections 2 and 3 of our previous work [8].

In the context of gene expression data, involved variables are often assumed to be multivariate normal distributed yielding to CI tests on the basis of the partial correlations [7]. While a single CI test is computational feasible, the total number of CI tests increases, in the worst case, exponential with regard to the number of variables in the dataset [9]. Under the reasonable assumption that the underlying causal graphical model is sparse, the complexity is reduced to be polynomial [3]. Yet, the long execution time, in particular for the skeleton discovery, remains a challenge for the application of the algorithm in practice [4].

In recent work, the parallel compute capabilities of GPUs have been utilized to significantly reduce the execution time [8, 10]. This approach is limited to datasets that fit entirely into the on-chip memory of modern GPUs, including necessary scratch space. Devices, such as the NVIDIA K80 GPU, provided by the Future SOC Lab, are equipped with Tesla GK210 cards having 12 GB of on-chip memory [5].

With our work, we address this limitation and propose an adaption of the PC-stable algorithm [1]. The algorithm is designed to be executed on the GPU, ensuring that all necessary CI tests are conducted, even if the dataset does not fit into the device memory. Additionally, it allows to distribute the computation to multiple GPUs.

2 A Skeleton Discovery Beyond GPU Memory Capacity

The skeleton discovery algorithm operates on the following data structures. For a set of N variables $\mathbf{V} = \{V_1, \dots, V_N\}$ it requires a correlation matrix, Cor, of dimension $N \times N$, two adjacency matrices, $adj_{in,out}(C, V_i)$, with C representing the undirected skeleton graph, each with a dimension of $N \times N$ and a data structure for the separation sets of dimension $N \times N \times l$, where l is the current level of the skeleton discovery. Hence, the required memory increases quadratic with N and slightly with each level l that is reached, e.g., due to dense causal graphical models. Once the on-device memory cannot hold all data structures, current implementations [8, 10] fail. In order to enable processing of datasets with a larger memory footprint, it is necessary to split the data into smaller blocks that fit into the on-device memory.

These smaller data blocks b have the following two requirements. First, each block contains the corresponding subsets of the above mentioned data structures. Second, the memory consumed by a single block has an upper limit mem_{block}, which is a fraction of the available device memory mem_{device}. The value of mem_{block} is restricted by the number of blocks required for the CI tests of the current level l. In addition, it is advisable that additional memory is reserved to allow overlap of data transfer and execution, a block b provides sufficient work to fully occupy the compute cores of the GPU and the number of CI tests are evenly distributed across blocks, for an extension to multiple GPUs.

Working on smaller data blocks b requires to adjust the skeleton discovery of the PC-stable algorithm. The result is a block-wise version, which we present in Algorithm 1. In the following, we highlight our changes to the original algorithm.

Algorithm 1: Block-wise skeleton discovery of PC-stable algorithm
Input: Vertex set V, correlation matrix Cor
Output: Estimated skeleton C, separation sets Sepset

1: Start with fully connected skeleton C and $l = -1$
2: **repeat**
3: $l = l + 1$
4: $updateAdjacency(adj_{in,out})$
5: $blocks = Split(\text{Cor}, adj_{in,out}, \text{Sepset}, l)$
6: **for all** b in $blocks$ **do**
7: Copy b to GPU
8: **if** $l == 0$ **then**
9: $BlockwiseCITests(b)$
10: **else**
11: $sepsetblocks = SepSetCombinations(b, l, blocks)$
12: **for all** s in $sepsetblocks$ **do**
13: Copy s to GPU
14: $BlockwiseCITests(b, s)$
15: **end for**
16: **end if**
17: Copy b from GPU
18: **end for**
19: $combineBlocks(blocks)$
20: **until** each adjacent pair V_i, V_j in C satisfy $|adj_{out}(V_i) \setminus \{V_j\}| \geq l$
21: **return** C, Sepset

The `Split()` operation in line 5 splits the input data structures and the adjacency lists into blocks. Therefore, it first calculates the upper limit mem_{block} for the current level l by $mem_{block} = \frac{mem_{device}}{(l+1)*(l+2)}$, note that memory for overlapping data transfer and execution is reserved in this calculation. Second, it chooses a suitable size $size_b$ for the blocks b in level l, with $size_b = \frac{N}{32 \cdot \mathsf{F}}$, following the constraint that the memory of b with $size_b$ does not exceed mem_{block}. Note, 32 is chosen, due to the warp size and F, with $\mathsf{F} \geq 1$ is a tuning parameter for the parallel work per block.

Next, the algorithm needs to calculate the CI tests for all blocks b. In level $l = 0$, see lines 8, 9, the CI tests are independent from any other edges in the skeleton C. Hence, all tests on edges within block b are conducted on data available in b. Therefore, the actual CI tests within the block b, conducted in function `BlockwiseCITests(b)`, can be executed in the same way as described in [8, 10].

In contrast, in levels $l \geq 1$ the CI tests are based on conditioning sets with a size of l. The conditioning sets are formed from the combinations of all adjacent nodes of V_i,V_j excluding V_i,V_j. Given a CI test on an edge in b it is no longer guaranteed that the data in b is sufficient to conduct the CI test. Therefore the algorithm requires additional blocks, the separation set blocks s, to process b, guaranteeing that each possible CI can be carried out, see lines $11 - 15$. The function `SepSetCombination()` provides a list of separation set combinations for the block b on the current level l. The level l determines the size of the separation set, while block b determines the candidate blocks for the combinations. The candidate blocks are all blocks that could contain nodes adjacent to any node in b. For example, given blocks with a dimension of 32×32 and a block b that contains the edges between $V_{32...64}$ and $V_{64...96}$. The candidate blocks are all blocks containing any of the edges between $V_{32...64}$ and V_j or between $V_{64...96}$ and V_j.

Next, for each separation set combination the according blocks are transferred to the device memory and a kernel is launched in line 14, which conducts the CI tests for block b given the separation set blocks s on the GPU. The kernel implementation for CI tests on higher level needs to be adapted to handle the separation of data into blocks. After all blocks have been processed they are combined into a single dataset again. This allows for an easier adjustment of the blocks in the subsequent level l.

3 Next Steps

In the current state, we have implemented the block-wise skeleton discovery for level $l = 0, 1$. An extension to the general case is planned after an experimental evaluation. For the experiments, we aim to determine a suitable size for the blocks b by varying the tuning parameter F, based on the memory capabilities of the K80 GPUs. The goal is to fully utilize the compute capacities, achieving minimum execution time. Furthermore, we plan to compare our block-wise algorithm that manages data transfer explicitly to the page migration engine on newer GPU-generations which allow to oversubscribe the GPU memory.

References

[1] D. Colombo and M. H. Maathuis. "Order-independent Constraint-based Causal Structure Learning". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pages 3741–3782. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2750365.

[2] A. P. Dawid. "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pages 1–31. ISSN: 0035-9246. URL: http://www.jstor.org/stable/2984718.

[3] M. Kalisch and P. Bühlmann. "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm". In: *J. Mach. Learn. Res.* 8 (May 2007), pages 613–636. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1248659.1248681.

[4] T. Le, T. Hoang, J. Li, L. Liu, H. Liu, and S. Hu. "A fast PC algorithm for high dimensional causal discovery with multi-core PCs". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (Feb. 2015).

[5] NVIDIA Corporation. *NVIDIA Tesla K80 The World's fastest GPU Accelerator*. Nov. 2014.

[6] J. Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 978-0-521-89560-6.

[7] A. Rau, F. Jaffrézic, and G. Nuel. "Joint estimation of causal effects from observational and intervention gene expression data". In: *BMC systems biology* 7.1 (Oct. 2013), page 111. DOI: 10.1186/1752-0509-7-111.

[8] C. Schmidt, J. Huegle, and M. Uflacker. "Order-independent Constraint-based Causal Structure Learning for Gaussian Distribution Models Using GPUs". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. SSDBM '18. Bozen-Bolzano, Italy: ACM, 2018, 19:1–19:10. ISBN: 978-1-4503-6505-5. DOI: 10.1145/3221269.3221292.

[9] P. Spirtes. "Introduction to Causal Inference". In: *J. Mach. Learn. Res.* 11 (Aug. 2010), pages 1643–1662. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1859905.

[10] B. Zare, F. Jafarinejad, M. Hashemi, and S. Salehkaleybar. "cuPC: CUDA-based Parallel PC Algorithm for Causal Structure Learning on GPU". In: *IEEE Transactions on Parallel and Distributed Systems* (2018). arXiv: 1812.08491 [cs.DC]. Forthcoming.

Deep Representation Learning on Large Attributed Graphs

Debora Nozza[1] and Enza Messina[1]

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano Bicocca
{debora.nozza,messina}@unimib.it

Attributed graphs convey a rich set of information on relations, nodes, and attributes. However, the entire set of information cannot be easily captured by traditional Representation Learning methods usually adopted for creating low-dimension and meaningful embeddings of the graph. This research project aims to generate a representation that encodes both the relational structure of a graph and the node attributes that can be texts or images. The final goal is to obtain an unsupervised model based on promising Deep Learning architectures that is able to efficiently and effectively derive dense vector representations of nodes in attributed graphs.

1 Introduction

One of the most crucial phase when addressing a machine learning problem is the definition of data representation. The input representation can considerably influence the performance of machine learning models, depending on its ability to disentangle and discover explanatory factors of variations behind the data given as input. A common practice is to exploit a-priori knowledge in order to design an ad-hoc representation depending on the domain, task or application. Although this feature engineering usually leads to improved results, it has several drawbacks: it requires the need of a domain expert resulting in a labor-intensive and not generalizable effort and it considers the data as identically distributed (i.i.d. assumption).

Representation Learning has become an important research field with the aim of providing novel methods for learning representation of the data that make it easier to extract useful information when building classifiers [14]. The interest in this field has recently grown because of the advent of Deep Learning and the ability to deal with unsupervised and semi-supervised learning.

The majority of the research contributions on Deep Learning focus on efficiently learning good representation for i.i.d. data [6, 8]. However, data can be represented in various forms and, in particular, relational structures are common representations used in many real-world problems, e.g. airline networks, publication networks, social and communication networks, and the World Wide Web.

Dealing with relational structures, such as graphs, is very complex and computationally expensive because of different characteristics of the data, i.e. size, dynamic nature, noise, and heterogeneity [3]. One efficient approach for handling potentially large and complex graph is to learn the graph representations, or Graph Embeddings [2, 4], which assign to each node of the graph a low-dimensional dense vector

representation, encoding meaningful information conveyed by the graph. Once the embedded representation is obtained, a wide variety of data mining problems can be solved by applying off-the-shelf algorithms designed for handling vector representations.

Several graph representation learning approaches at the state of the art focus only on the graph structure to compute the graph embeddings [9, 11, 13]. However, nodes in real-world graphs are often associated with a rich set of features or attributes (e.g. text, image, audio), originating the so-called attributed graph.

Although attributed graphs convey a rich set of information on relations, nodes, and attributes, they cannot be easily given as input to classic representation learning methods. In order to overcome this limitation, the main goal of this project is the generation of a representation able to encode both the node attributes and the graph structure. Indeed, the consideration of both information could strongly improve the meaning encoded in the representation and consequently the performance of the graph mining tasks.

2 Problem

The research project aims at designing and developing novel unsupervised models for learning a graph representation from large heterogeneous attributed graphs, which comprises both the structure of the graph and the attributes associated with each node. The proposed graph representation learning model will be based on deep learning models, strengthened by an efficient optimization algorithm able to scale for large graphs.

The proposed research project will face the following challenges:

1. Structure-preserving: graph embeddings should preserve the structure of the graph, which is often complex and highly non-linear. Moreover, how to simultaneously preserve the local and global structure is also a tough problem.

2. Scalability: most real-world graphs are huge and contain millions of nodes and edges. The graph representation learning model should be scalable and able to process large graphs.

3. Sparsity: many real-world graphs are often so sparse that considering only (few) observed links is not enough to reach a satisfactory performance [9].

4. Dimensionality of the embedding: the dimension of the embedded representation should be chosen as a trade-off between reconstruction precision performances and time and space complexity. The choice can also be application-specific depending on the objective task.

5. Attribute expression: the obtained graph embedding should be able to directly encode the attribute features in addition to the graph structure information.

3 Implementation

The research project will be implemented in Python, by taking advantage of the Keras[1] library for Deep Learning. Keras is a minimalist, highly modular neural networks library written in Python and capable of running on top of either TensorFlow or Theano. The Keras Deep Learning library permits to:

- Easy and fast prototype, through user friendliness, modularity, and extensibility.
- Support several Deep Learning architectures, such as convolutional and recurrent networks.
- Run code on CPU and GPU.

Since large attributed graphs, which is the input structure that we want to investigate, are commonly associated with millions of nodes and related attributes [12], they require a large amount of memory to be loaded. Moreover, the use a GPUs would permit to substantially scale and more efficiently deal with the training process of Deep Learning models on large amount of data. We gratefully acknowledge the support of the HPI Future SOC Lab, for providing the IT infrastructure and the access to NVIDIA Tesla K80 that permits the realization of this project.

4 Evaluation

The evaluation will be performed on several graph mining tasks, i.e. graph reconstruction, link prediction, node classification, clustering and visualization. The analysis will be conducted on real datasets originated from different domains, such as social network, blogs, scientific collaboration networks, and biological interaction networks.

5 Related Work

Over the past decade, graph representation learning has attracted a surge of research attention, particularly focused on developing new embedding algorithms. In this domain, research studies can be roughly distinguished in factorization methods and deep learning approaches. The basic idea underlying both methods is to preserve both the local and global graph structure in the embedded vector space.

Factorization based algorithms represent the connections between nodes in the form of a matrix and factorize this matrix to obtain the embeddings [1, 2, 7, 10, 11]. The growing interest in deep learning algorithms has affected also the task of graph

[1] https://keras.io/ (last accessed 2020-04-01).

representation learning due to their ability to model non-linear structures in the data. Beyond Deep Learning methods applied on random walks [9], most of the investigations are focused on improving deep learning architectures [13].

Only a few studies considered the set of features associated to each node in addition to the graph topological structure (attributed graph) [3, 5]. Their limitation regards the fact that they do not explicitly consider the nodes' attributes but only a measure of attribute similarity between them, resulting in a lower representation expressiveness.

6 Conclusion

The research project will aim to propose a novel unsupervised model for attributed graph embeddings. It is expected that considering the relational information in addition to the attribute information will provide significant improvements.

Future work will be focused on dealing with attributed graphs with probabilistic relationships. Creating meaningful embeddings of attributed graphs where noisy and uncertain relations are considered represents a major challenge for tackling real word problems.

References

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. "Distributed Large-scale Natural Graph Factorization". In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pages 37–48.

[2] M. Belkin and P. Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". In: *Neural Computation* 15.6 (2003), pages 1373–1396.

[3] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. "Heterogeneous Network Embedding via Deep Architectures". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pages 119–128.

[4] P. Goyal and E. Ferrara. "Graph Embedding Techniques, Applications, and Performance: A Survey". In: *Knowledge-Based Systems* 151 (2018), pages 78–94. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.03.022. arXiv: 1705.02801 [cs.SI].

[5] X. Huang, J. Li, and X. Hu. "Accelerated Attributed Network Embedding". In: *Proceedings of the 2017 SIAM International Conference on Data Mining*. 2017, pages 633–641.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* (2013). arXiv: 1301.3781 [cs.CL].

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. "Asymmetric Transitivity Preserving Graph Embedding". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pages 1105–1114.

[8] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Volume 14. 2014, pages 1532–1543.

[9] B. Perozzi, R. Al-Rfou, and S. Skiena. "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pages 701–710.

[10] S. T. Roweis and L. K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500 (2000), pages 2323–2326.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. "LINE: Large-scale Information Network Embedding". In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pages 1067–1077.

[12] L. Tang and H. Liu. "Scalable learning of collective behavior based on sparse social dimensions". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pages 1107–1116.

[13] D. Wang, P. Cui, and W. Zhu. "Structural Deep Network Embedding". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pages 1225–1234.

[14] A. C. Yoshua Bengio and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* 35.8 (2013), pages 1798–1828.

Measurement-Based Software Performance Engineering for Microservices and Multi-Core Systems

André van Hoorn, Markus Frank, and Henning Schulz

Institute of Software Technology
University of Stuttgart

This report provides a summary of our project "Measurement-Based Software Performance Engineering for Microservices and Multi-Core Systems" conducted during the HPI Future SOC Lab period fall 2018, as well as ideas for a follow-up project for the upcoming period.

1 Project Idea

Our project was divided into two subprojects, namely (1) "DevOps-oriented Load Testing for Microservices" and (2) "Software Performance Engineering for Multi-Core Systems". Subproject (1) is a direct continuation of the works that we started in the previous periods. Subproject (2) is new and we had started initial experiments on the HPI infrastructure in the past Future SOC Lab period. For both subprojects, in order to conduct large-scale experimental evaluations, we need a state-of-the-art computing infrastructure such as the one provided by the HPI Future SOC Lab.

In the remainder of this report, we will provide some more details about the project context (Sections 1.1 and 1.2) list the granted Future SOC Lab resources (Section 2), provide a brief description of our findings (Section 3), and outline next steps (Section 4).

1.1 DevOps-oriented Load Testing for Microservices

Modern software engineering paradigms and technologies — such as DevOps [2] (including automation as part of continuous delivery) and microservices [12] — are gaining more and more attraction in the software and services engineering communities. Of particular interest are quality-of-service concerns, for instance, w. r. t. performance and reliability. While established approaches for classic contexts (i.e., which do not use DevOps and microservices) exist, their adoption to DevOps and microservices requires considerable research efforts [3, 10].

In the recent years, our group has already contributed architecture-aware approaches for performance and reliability, involving a combination of measurement-based and model-based techniques [11, 13, 15]. Recently, we started to investigate how these techniques can be used in DevOps and microservice contexts — with a particular focus on load testing as a key performance engineering activity [4, 14, 15].

In the past period, we have published a conference paper [1] including measurements conducted in the HPI infrastructure. First of all, the activities on load testing

conducted during this period were a direct continuation of the activities started during the previous periods. The activities were planned for the following topics:

1. Detection of performance regressions based on load tests
2. Prioritization and selection of load tests
3. Advanced extraction of load test specifications from APM data

1.2 Software Performance Engineering for Multi-Core Systems

Multicore systems are a permanent part of our daily life. Regardless of whether we consider nowaday's desktop PCs, notebooks, or smart phones—all devices are running on multicore CPUs. To use these hardware features in an efficient way, developers need to build parallel-enabled software. However, the development of such software is more complex than developing sequential software.

To handle the rising complexity, it is necessary to develop software in an engineering-like way. In such a process, software architects plan and analyze software designs on a model level. Software architects can use tools like Palladio to simulate and analyze early-phase software designs. Unfortunately, current approaches and tools lack the ability to consider multicore systems. Therefore, in this project, we aim to find performance prediction methods for multicore systems in the context of our ongoing research [6, 7, 8, 9].

In the previous period, we have started to use the HPI infrastructure to study performance properties for performance predictions of multi-core systems. The plan for this period was to conduct further experiments with different configurations (e.g., thread numbers and thread pool sizes) to assess what is their impact on performance properties for different use cases or scenarios (e.g., benchmarks) to obtain performance curves. These performance curves will be used by software architects to easily adopt performance prediction models.

2 Used Future SOC Lab resources

We requested and received dedicated (root) access to the following computing resources (servers): *i.*) 896 GB RAM, 80 cores; *ii.*) 32 GB RAM, 24 cores. Dedicated access has been given to us due to our expected high resource demands.

3 Findings

We have worked on the previously stated goals in both subprojects. Not for all of them, we have conducted experiments in the HPI Future SOC Lab, yet. Additional experiments are planned for the next period. In this section, we will provide a summary of the experiments and results for both subprojects.

3.1 DevOps-oriented Load Testing for Microservices

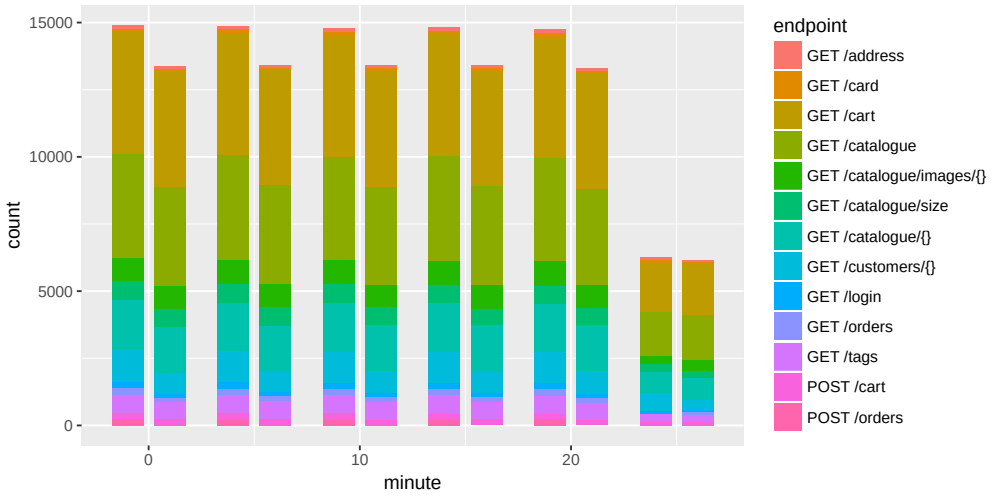


Figure 1: Request counts per 5 minutes of the reference workload (left bars) and the generated workload (right bars)

We investigated whether we can generate a load test for a microservice application that represents the workload in the production system. For this, we execute several experiments with the Sock Shop microservices demo application.[1] We deployed the Sock Shop application on the server *i.*) and a load driver on the server *ii.*), as listed in Section 2. While executing a load test against the Sock Shop for simulating production usage, we monitor the Sock Shop application and collect traces. Based on the traces, we generate a new load test targeting the microservices of the Sock Shop and executed it as well.

Figure 1 provides preliminary results of this experiment. The left bars of the respective groups represent the number of requests submitted per 5 minutes and per endpoint of the Sock Shop. The right bars represent the same information for the generated test. It can be seen that the endpoint ratio is generally accurate, but the generated test submits slightly less requests overall. We are currently improving our approach to resolve this drawback and also investigate advantages of different types of load test generations.

3.2 Software Performance Engineering for Multi-Core Systems

We have investigated the impact of different system configurations on the execution behavior of parallel programs. The goal is to use these measurements to improve cur-

[1] https://microservices-demo.github.io/ (last accessed 2020-04-01).

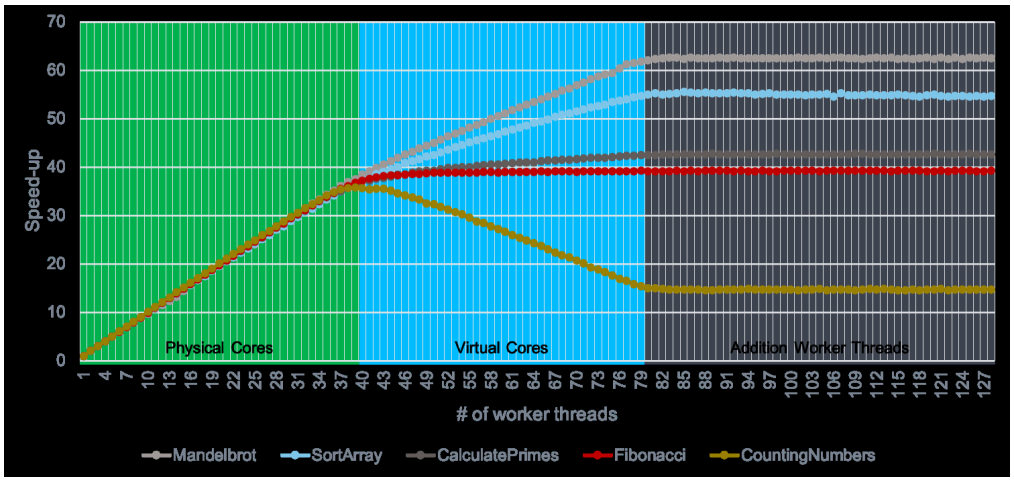


Figure 2: Measurements of speedup functions for different resource demands on the 40-core system with enabled hyper-threading [5]

rent performance prediction models by providing reference curves for the speedup behaviour of different resource demands and scenarios.

We used synthetic programs (processor-intensive and I/O intensive) with generated resource demands and executed them in series of experiments using different configurations, numbers of threads, software caches, and data localities. Figure 2 shows the speedup curves for the individual programs by using a different number of worker threads. Each data point represents speedup compared to sequential execution.

A paper including experiments conducted in the HPI Future SOC Lab has been accepted for the 10th ACM/SPEC International Conference on Performance Engineering (ICPE 2019) [5]. We had presented preliminary results at the HPI Future SOC Lab Day in fall 2018.

4 Next steps

We will continue our ongoing research in the two subprojects. To have the ability to execute extensive experiments, we will apply for the next HPI Future SOC Lab period. We plan to extend our load testing approaches to Functions-as-a-Service (FaaS). Extensions for the multi-core experiments include different configurations and more complex parallel programs.

Acknowledgment

We are thankful to the HPI Future SOC Lab for having granted us access to the computing infrastructure. The environment eases the joint work of different organi-

zations on the platform, which has so far been hindered by university-internal access constraints—apart from the fact that an equipment comparable to that of the HPI Future SOC Lab has not been available to us, and that enables extensive performance configuration tests needed to reach our goals.

References

[1] A. Avritzer, V. Ferme, A. Janes, B. Russo, H. Schulz, and A. van Hoorn. "A quantitative approach for the assessment of microservice architecture deployment alternatives using automated performance testing". In: *Proceedings of the 12th European Conference on Software Architecture (ECSA 2018)*. LNCS. Springer, 2018.

[2] L. J. Bass, I. M. Weber, and L. Zhu. *DevOps — A Software Architect's Perspective*. SEI series in software engineering. Addison-Wesley, 2015. ISBN: 978-0-13-404984-7.

[3] A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. Herbst, P. Jamshidi, R. Jung, J. von Kistowski, A. Koziolek, J. Kroß, S. Spinner, C. Vögele, J. Walter, and A. Wert. *Performance-oriented DevOps: A Research Agenda*. Technical report SPEC-RG-2015-01. SPEC Research Group — DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), 2015.

[4] V. Ferme and C. Pautasso. "Integrating Faban with Docker for Performance Benchmarking". In: *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016)*. ACM, 2016.

[5] M. Frank, S. Becker, A. Kaplan, and A. Koziolek. "Performance-influencing Factors for Parallel and Algorithmic Problems in Multicore Environments". In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19)*. To appear.

[6] M. Frank and M. Hilbrich. "Performance Prediction for Multicore Environments — An Experiment Report". In: *Proceedings of the Symposium on Software Performance (SSP 2016)*. 2016.

[7] M. Frank, M. Hilbrich, S. Lehrig, and S. Becker. "Parallelization, Modeling, and Performance Prediction in the Multi-/Many Core Area: A Systematic Literature Review". In: *Proceedings of the 7th IEEE International Symposium on Cloud and Service Computing (SC2 2017)*. 2017.

[8] M. Frank, F. Klinaku, and S. Becker. "Challenges in Multicore Performance Predictions". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE 2018)*. ACM, 2018.

[9] M. Frank, S. Staude, and M. Hilbrich. "Is the PCM Ready for ACTORs and Multicore CPUs? — A Use Case-based Evaluation". In: *Proceedings of the 8th Symposium on Software Performance (SSP 2017)*. 2017.

[10] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger. "Performance Engineering for Microservices: Research Challenges and Directions". In: *Companion of the 8th ACM/SPEC International Conference on Performance Engineering* (*ICPE 2017*). ACM, 2017.

[11] A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Kiel Computer Science Series 2014/6. Dissertation, Faculty of Engineering, Kiel University. Kiel, Germany: Department of Computer Science, Kiel University, 2014.

[12] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[13] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske. "Hora: Architecture-aware online failure prediction". In: *Journal of Systems and Software* (2018). DOI: 10.1016/j.jss.2017.02.041.

[14] H. Schulz, T. Angerstein, and A. van Hoorn. "Towards Automating Representative Load Testing in Continuous Software Engineering". In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (*ICPE 2018*) *Companion* (*7th International Workshop on Load Testing and Benchmarking of Software Systems, LTB 2018*). ACM, 2018.

[15] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, and H. Krcmar. "WESSBAS: Extraction of Probabilistic Workload Specifications for Load Testing and Performance Prediction—A Model-Driven Approach for Session-Based Application Systems". In: *Journal on Software and System Modeling* (*SoSyM*) (2018).

Collecting More Tweets From Twitter API

A Case Study of Twitter's Snowflake Algorithm for Generating Unique IDs

Seyed Ali Alhosseini and Christoph Meinel

Internet Technologies and Systems
Hasso Plattner Institute for Digital Engineering
{seyedali.alhosseni,christoph.meinel}@hpi.de

Twitter as an online social network is a rich platform of user generated content. The Twitter API gives access to tweets posted by users. In this paper we analyze the unique tweet ID numbers over a collection of 10 million tweets. The results show that 60 % of tweets have an identical sequence id and 95 % tweets are located in two data centers. Based on our findings we show how we can collect a higher sample of tweets in a more efficient approach.

1 Introduction

There are many online social platforms for people to express themselves, share their thoughts and interact with one another. Thus, online social networks have become a massive source of information spreading and diffusion. In the past decade we have seen a huge growth of user generated content on online social networks. Twitter as a microblogging service has become a medium for users to share information and news [4]. Various research groups have collected this data for topic modeling, community detection, user personality and behavior analysis etc. However collecting data from Twitter's API is limited to a 1 % sample of all the tweets. We show it possible to collect a higher sample of tweets based on the Snowflake algorithm and the format of tweet ids.

In this study, we investigate the following questions:

1. What format is Twitter's Snowflake method using to generate unique ids?
2. How can we collect a bigger sample of tweets based on the tweet id format?
3. With how many API keys can we collect a higher sample of tweets from Twitter's timeline?

2 Background

2.1 Twitter API

The Twitter Application Programming Interface (API) has seen several changes since the microblogging website was first launched in 2006. Before 2009, it was possible for research groups to crawl tweets on Twitter at a large scale. For example [3] collected 106 million tweets and 41.7 million user profiles. However in September 2009, Twitter announced a new terms of service with a new rate-limit for the API [7]. Depending on the API function an authorized user can make a specific number of requests in a 15 minutes window. For example for collecting followers' ids 15 requests and for getting users' profile 900 requests can be made before you hit the rate limit.

2.2 Snowflake

In June 2010 Twitter introduced Snowflake as their new approach for generating unique ID numbers for tweets [2]. Snowflake addresses the following challenges:

1. How to generate ids in the scale of tens of thousands per second,
2. How to generate ids in a distributed and uncoordinated approach,
3. How to generate ids so they are roughly sortable by time.

3 Tweet ID

In this section we take look at the unique 64-bit unsigned integer ids in the tweet object. The Snowflake project is open source and the code for generating ids is available online [6]. We can see from the IdWorker file code, how a tweet id is generated. A tweet id consists of a timestamp, datacenterId, workerId and sequenceId. The timestamp has an offset of 1288834974657L (Thursday, November 4, 2010 1:42:54.657 AM) which is when snowflake first started. Figure 1 shows the tweet ID format.

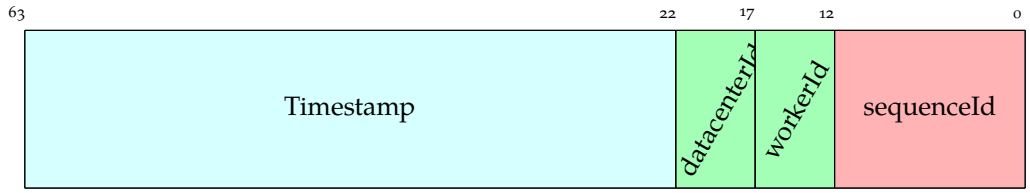


Figure 1: Tweet ID Binary Format

Listing 1: IdWorker.scala

```
...
val twepoch = 1288834974657L

val workerIdBits = 5L
val datacenterIdBits = 5L
...
val sequenceBits = 12L

val workerIdShift = sequenceBits
val datacenterIdShift = sequenceBits + workerIdBits
val timestampLeftShift = sequenceBits + workerIdBits +
    ↪ datacenterIdBits
...
def nextId(): Long = synchronized {
...
((timestamp - twepoch) << timestampLeftShift) |
    (datacenterId << datacenterIdShift) |
    (workerId << workerIdShift) |
    sequence
}
```

4 Experiment

4.1 Resources from HPI Future SOC Lab

In this project, we had access to a VM with 8 GB RAM and 8 CPU cores. We collected and stored the tweets using python. We analyzed the 45 GB tweet ids with Apache Spark.

4.2 Collecting Tweets from statuses/sample

The statuses/sample API function returns a small random sample of all public statuses via a stream [1]. We collected 10 million tweets from March 1st to March 3th, 2019 using this API function. On average 3320 tweets were collected per minute. However using this approach gives 1 % of Twitter's timeline [5]. Connecting with multiple API keys will not result in a higher sample of the tweets in the timeline. Because as mentioned in the Twitter API documentation if two different clients connect to this endpoint, they will see the same tweets [1]. In the next section we see how we can collect a higher sample based on the tweet id format.

4.3 Collecting Data from statuses/lookup

The statuses/lookup API function takes 100 tweet ids as input and returns the tweets in JSON format. With a developer API key, it is possible to make 1200 requests in a 15 minutes window. 900 requests as an authorized user and 300 as an authorized app. Therefore it is possible to check 120,000 tweet ids every 15 minutes.

4.4 Case 1: Scanning all the space for 1 minute of the Twitter timeline

In the first case we compute the amount of time necessary to collect 1 minute or 60,000 milliseconds of all the tweets on Twitter with one developer key. As shown in section 3, 22 bits are available for the datacenterId, workerId and sequenceId. So the total number of ids that can be generated is:

$$2^{22} = 4,194,304 \tag{1}$$

With one API key the time necessary to check all the generated ids for 1 minute is:

$$1 \times 60 \times 1000 \times 4,194,304 = 251,658,240,000$$

$$\lceil \frac{251658240000}{120000} \rceil = 2097152$$
$$2097152 \times 15 = 31457280\, minutes \approx 60\, years$$

Clearly scanning the entire 22 bits space to collect 1 minute is not a feasible solution. However, based on our findings in table 2 to table 3, it is not required to consider all the possible combinations. In the next case we see how we can reduce the number of ids to check.

4.5 Case 2: Collecting 1 minute of the Twitter timeline with 90 % sampling and single API key

In this scenario we compute the amount of time necessary to collect 90 % of tweets in 1 minute of the Twitter timeline with one developer key. From table 2 we select the first 10 sequenceIds which result in 99.526 % tweets. These sequence ids include {0, 1, 2, 5, 3, 6, 7, 4, 8, 10}. From table 3 we select the first 25 datacenterId+workerId which will result in 90.561 % of the tweets. The selected ids for the datacenter and worker include {363, 365, 375, 336, 335, 361, 382, 376, 333, 332, 350, 381, 364, 327, 326, 342, 362, 372, 347, 334, 35, 36, 33, 34}. By selecting these ids we can collect tweets by a sample of 90 %:

$$0.90561 \times 0.99526 \approx 90.13\,\%$$

$$1 \times 60 \times 1000 \times 25 \times 10 = 15,000,000$$

$$\lceil \frac{15000000}{120000} \rceil = 125$$
$$125 \times 15 = 1875\, minutes = 31\, hours\, 15\, minutes$$

Compared to case 1 this approach is more tractable. In the next case we see how we can reduce the time with multiple API keys.

4.6 Case 3: Collecting in read-time the Twitter timeline with a 90 % sample and Multiple API keys

Based on the previous case, we are interested to find the number of API keys required to collect a sample of 90 % of the tweets in real-time. For each extra authorized user we are able to check 90,000 more ids.

$$15 \times 60 \times 1000 \times 25 \times 10 = 22,500,000$$

$$\lceil \frac{22500000}{120000 + 90000 \times n} \rceil = 1$$
$$n \approx 2499$$

Therefore with 2500 API keys it is possible to collect 90 % of the tweets in real-time.

5 Results

In this section, we report the results. Table 1 shows the distribution of tweet ids based on the 5 bit representing the data center id. 95 % of the tweets are stored in 2 data centers with ids 11, 10.

Table 1: DatacenterId

DatacenterId	binary	count	Percentage	Cumulative Percentage
11	01011	4 786 459	47.865	47.865
10	01010	4 714 581	47.146	95.010
1	00001	356 914	3.569	98.580
4	00100	125 841	1.258	99.838
0	00000	6637	0.066	99.904
24	11000	394	0.004	99.908
12	01100	388	0.004	99.912
19	10011	383	0.004	99.916
27	11011	377	0.004	99.920
29	11101	368	0.004	99.923

Table 2 shows the distribution of tweet ids based on the 12 bit representing the sequence id. Around 60 % of tweets have sequence id of zero (0).

Table 2: Sequence Id

sequenceid (12 bits)	count	Percentage	Cumulative Percentage
0	5957392	59.574	59.574
1	2416262	24.163	83.737
2	786100	7.861	91.598
5	255150	2.551	94.149
3	215668	2.157	96.306
6	164613	1.646	97.952
7	69542	0.695	98.647
4	64925	0.649	99.297
8	22962	0.230	99.526
10	11559	0.116	99.642
11	8802	0.088	99.730
9	6701	0.067	99.797
12	4287	0.043	99.840
13	1665	0.017	99.856
15	597	0.006	99.862
14	515	0.005	99.867
16	480	0.005	99.872
17	262	0.003	99.875
32	154	0.002	99.876
18	135	0.001	99.878

6 Conclusion and Future Work

In this paper we studied Twitter's Snowflake algorithm for generating unique tweet IDs. The Twitter API is limited to a small sample of tweets. We show that based on the binary representation of tweet ids and with access to multiple API keys it is possible to circumvent this limitation.

In future work, our goal is to develop an online platform where users can authorize their Twitter account and can get access to higher sample of tweets in real-time.

Table 3: DatacenterId+workerId

DatacenterId+workerId	binary	count	Percentage	Cumulative Percentage
363	0101101011	463 652	4.637	4.637
365	0101101101	459 792	4.598	9.234
375	0101110111	458 079	4.581	13.815
336	0101010000	457 315	4.573	18.388
335	0101001111	447 995	4.480	22.868
361	0101101001	447 860	4.479	27.347
382	0101111110	447 300	4.473	31.820
376	0101111000	446 723	4.467	36.287
333	0101001101	446 495	4.465	40.752
332	0101001100	444 892	4.449	45.201
350	0101011110	444 542	4.445	49.646
381	0101111101	443 513	4.435	54.082
364	0101101100	441 847	4.418	58.500
327	0101000111	435 805	4.358	62.858
326	0101000110	433 706	4.337	67.195
342	0101010110	431 715	4.317	71.512
362	0101101010	431 688	4.317	75.829
372	0101110100	425 535	4.255	80.085
347	0101011011	425 534	4.255	84.340
334	0101001110	408 800	4.088	88.428
35	0000100011	53 422	0.534	88.962
36	0000100100	53 341	0.533	89.496
33	0000100001	53 315	0.533	90.029
34	0000100010	53 232	0.532	90.561
32	0000100000	53 229	0.532	91.093
325	0101000101	39 727	0.397	91.491
352	0101100000	35 527	0.355	91.846
366	0101101110	35 147	0.351	92.197
373	0101110101	32 149	0.321	92.519
323	0101000011	31 398	0.314	92.833

References

[1] *GET statuses/sample — Twitter Developers*. 2010. URL: https://developer.twitter.com/en/docs/tweets/sample-realtime/api-reference/get-statuses-sample (last accessed 2019-03-01).

[2] R. King. *Announcing Snowflake - Twitter Blog*. 2010. URL: https://blog.twitter.com/engineering/en_us/a/2010/announcing-snowflake.html (last accessed 2019-02-28).

[3] H. Kwak, C. Lee, H. Park, and S. Moon. "What is Twitter, a Social Network or a News Media?" In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pages 591–600. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772751.

[4] C. Meinel, J. F. M. Broß, P. Berger, and P. Hennig. *Blogosphere and its Exploration*. Springer, 2015. ISBN: 978-3-662-44408-5. DOI: 10.1007/978-3-662-44409-2.

[5] F. Morstatter, J. Pfeffer, H. Liu, and K. Carley. "Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose". In: *International AAAI Conference on Web and Social Media*. 2013. DOI: 10.1609/icwsm.v7i1.14401.

[6] *Release snowflake-2010 · twitter-archive/snowflake*. 2010. URL: https://github.com/twitter-archive/snowflake/releases/tag/snowflake-2010 (last accessed 2019-03-01).

[7] B. Stone. *Twitter's New Terms of Service*. 2009. URL: https://blog.twitter.com/en_us/a/2009/twitters-new-terms-of-service.html (last accessed 2019-03-07).

Benchmarking FFT on an Ethernet Cluster

Latency, Bandwidth and MPI benchmarks

Marek Nowicki[1], Benson K. Muite[2], and Mehmet Can Boysan[2]

[1] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
faramir@mat.umk.pl
[2] Institute of Computer Science
The University of Tartu
{benson.muite,mehmet.can.boysan}@ut.ee

The performance of Ethernet enabled communications is measured on the HPI 1000 core cluster for communication intensive workloads, in particular the Fast Fourier Transform.

1 Introduction

Ethernet clusters are used in a variety of parallel computing clusters because the technology is well known and there are a number of software frameworks built on top of ethernet, which may not be supported on infiniband and other high performance communication interfaces. The purpose of this study is to measure the performance of the 1000 core HPI cluster that uses 10 gigabit ethernet.

Contributions

The main findings are that further tuning of and configuration for the 1000 core cluster is required to obtain good communication performance on the current hardware. The default installations of MPI and SLURM job manager are from the Ubuntu repositories. These do not give optimal performance. By building and installing OpenMPI 4.0, a significant performance boost in communication is obtained. Default setups of MPICH and MVAPICH are typically not as performant as OpenMPI for the considered tests, but they may further enable performance improvements for jobs that utilize multiple GPUs. Such improvements would likely require reinstallation of the SLURM job management system with updated PMI support. The performance of MPI is also compared with PCJ, a framework for parallel computing in Java that uses sockets for communication.

2 Context

In a previous study, Aseeri et al. found that a 10 Gigabit ethernet cluster provided good performance on upto 100 cores for a communication intensive Fast Fourier transform based program for solving the Klein Gordon equation. The cluster, Neser,

is no longer operational and the full details of the setup of that cluster are unknown. It is of interest to determine whether other 10 Gigabit ethernet clusters can be as performant.

Background

Parallel and big data computing require highly configurable and performant software stacks. For many problems, low interconnect latency and high interconnect bandwidth enable greater performance in solving problems which have communication requirements. Programming patterns that have significant communication requirements include parallel graph traversal, fast Fourier Transforms. For large Fourier transforms, high interconnect bandwidth can be a limiting factor. For graph traversal and small Fourier transforms, latency is often the limiting factor. While the chosen hardware gives a lower bound to the latency and an upper bound to the bandwidth, the software stacks used for communication also have a significant effect on the realizable latency and bandwidth that will affect application performance.

3 Methodology

To measure bandwidth an latency on MPI enabled clusters, the OSU MPI and Intel MPI benchmarks are often used. These are a collection of tests to measure performance of MPI installations by making and timing specific MPI calls for a range of data sizes. They enable comparison of different clusters. Unfortunately, at present, there are no datasets with performance information for these benchmarks that would enable easy comparison of cluster performance. Both these MPI benchmarks are often used to monitor cluster health. In this case, only the PingPong, bandwidth and alltoall benchmarks are considered. In addition a new nonblocking PingPong benchmark which utilizes isend and irecv is written and tested. The benchmarks are tested with 4 libraries, OpenMPI 4.0.0, MPICH-3.3 , MVAPICH2-2.3.1 and PCJ5.0.8. To enable reproducibility, installation instructions and configuration of the libraries are in the appendices. Run data includes information about the nodes used during each execution. Job interference did not occur during executions because only a single job with inter processor communication was running at any one time. Each test was also run at least thrice, with the mean result, and the maximum and minimum results being reported, scripts to produce the executions are also included.

3.1 Hardware Description

The HPI 1000 core cluster is composed of Intel(R) Xeon(R) E7-4870 CPUs clocked at 2.40 GHz. Each node has 4 CPUs and a total of 40 cores, with hyperthreading enabled. Each node also has a two 10 Gigabyte ethernet Intel 82599ES adaptors and 1 TB of RAM.

3.2 Testing Environment

The cluster runs Ubuntu 16.04 and uses SLURM 17.02.6 - SLURM has been installed from the Ubuntu repositories. The compiler suite is GNU compilers version 5.4.0, with the GCC, G++ and GFORTRAN as the C, C++ and Fortran compilers respectively. Java Openjdk version "9-internal" from the Ubuntu repositories is the installed Java library.

There are four non-GPU partitions on the cluster with 10, 6, 4 and 2 nodes. In addition, there is a single node DGX partition with 8 Nvidia V100 GPUs, a single node partition with 2 Nvidia K20 GPUs, a single node partition with 4 Nvidia K80 GPUs and a dual node partition with 6 Nvidia K80 GPUs. All tests reported here were run on the 10 node partition.

The latest version of SLURM has MPI plugins for OpenMPI, PMI2, PMIX and an unoptimized default for any other MPI library[3] At present only the OpenMPI and default plugins are available on the HPI cluster.

There is an installed OpenMPI version 1.10.2 and MPICH 3.2 from Ubuntu repositories on the cluster which provides mpirun, mpicc, mpicxx and mpif90 functionality. The user needs to choose the appropriate set of compilers and runtime executables. In addition to the installed versions of OpenMPI and MPICH, OpenMPI 4.0.0, MPICH-3.3 and MVAPICH2-2.3.1 were installed using the following commands:

Listing 1: Installation for MVAPICH2-2.3.1

```
./configure --prefix=$HOME/mvapich2-2.3.1-install \
  --with-device=ch3:nemesis:tcp --disable-cuda
make
make install
```

Listing 2: Installation for MPICH-3.3

```
./configure --prefix=$HOME/mpich-3.3-install \
  --disable-cuda --with-pm=hydra
make
make install
```

[3] https://slurm.schedmd.com/mpiplugins.html (last accessed 2020-04-01).

Listing 3: Installation for MPICH-3.3

```
./configure --prefix=$HOME/openmpi-4.0.0-install \
  --enable-mpi-java
make
make install
```

4 Results

Descriptions of the benchmarks and results for running them on the HPI 1000 core cluster now follow. Each benchmark was run using several parallel environments to compare the performance. It should be noted that further tuning and optimization is possible for each of the environments, thus good or bad performance on one benchmark alone is not conclusive evidence of the quality and performance of the entire software for solving real world problems.

4.1 The PingPong Benchmark

The PingPong benchmark is a standard benchmark to test the latency between two processes. It measures the time for round trip information between the two processes as a function of the amount of information that is sent between the two processes. It gives a lower bound on the amount of time needed for work to occur below which parallelization will not produce any speedup. PingPong benchmarks are available in both the Intel MPI benchmark suite and in the Ohio State MPI benchmark suites. In addition, Boysan has written a Java based Ping Pong benchmark. The MPI implementations use blocking MPI_send and MPI_recv calls.

4.2 The Nonblocking PingPong Benchmark

PCJ typically relies on asynchronous communication for data exchange, in particular put and get calls to asynchronously send information to or pull information from another process. Thus the regular Ping-Pong benchmark using blocking MPI send and MPI recv calls is not the most appropriate one for comparison of performance between PCJ and MPI implementations. A better measure is a Ping Pong benchmark using nonblocking MPI_Isend and MPI_Irecv calls.

4.3 The Bandwidth Benchmark

In addition to latency, bandwidth is also an important system characteristic and gives an indication of how much information can be sent between two processes within a fixed unit of time, neglecting the time to start and stop data transmission. For programs that need to send a lot of data, this gives a lower bound on the amount

of time the communication phase of the program will require. Both the Intel and Ohio State MPI benchmarks include a point to point bandwidth benchmark.

4.4 The AllToAll Benchmark

An all to all exchange is required of many Fast Fourier transform programs. Depending on the amount of data being exchanged, it can either be latency bound or bandwidth bound. In comparison to the point to point latency and bandwidth benchmarks, an all to all exchange will typically show higher latency and lower bandwidth between any two points than the point to point benchmarks because of network congestion.

4.5 The 1D Fourier Transform Benchmark

The parallel one dimensional Fast Fourier Transform is a benchmark included in the HPCC challenge suite. It is a computation that is typically either limited by bandwidth from main memory, network bandwidth or network latency. Performance on this benchmark is measured by the number of floating point operations that can be done.

4.6 The 3D Fourier Transform Benchmark

Three dimensional Fast Fourier Transforms often are used in the numerical solution of partial differential equations. There are a number of libraries available for computing these, in particular 2DECOMP&FFT, FFTE, P3DFFT, P3DFFT++, ACCFFT, PFFT etc. To enable separation of concerns, many of these libraries use an optimized one dimensional FFT engine such as vendor provided libraries at the single core or node level. The library then does domain decomposition, typically either one dimensional (slab), or two dimensional (pencil). The library is also responsible for data transposition. FFTE is one of the few libraries where the author of the serial FFT and the parallel FFT are the same. In this work tests are done with 2DECOMP&FFT since the original aim was to see if other 10 Gigabit ethernet enabled clusters could have good performance. The three dimensional FFT is a benchmark problem in the NAS parallel benchmarks, but unfortunately, performance record histories for the NAS parallel benchmarks are not available.

4.7 The Klein Gordon Benchmark

Aseeri et al. compared the performance of several different supercomputers when solving the Klein Gordon equation using a Fourier spectral method. The code uses the library 2DECOMP&FFT in a semi-implicit time stepping scheme to solve

$$u_{tt} = u - u3 + \Delta u$$

approximated by

$$\frac{u_{n+1} - 2u_n + u_{n-1}}{(\delta t)^2} = \frac{\Delta - 1}{4}(u_{n+1} - 2u_n + u_{n-1}) + u_n^3$$

Time stepping occurs in Fourier space with a backward three dimensional FFT to calculate u in real space, a calculation of u_n^3 from u_n in real space, and then a forward FFT to get u_n^3 in Fourier space. In the benchmark a discretization of 512^3 points was used and the time to take 30 time steps was recorded for several different choices of processors—in particular on each platform, a strong scaling test was undertaken to find the platform with the fastest time to solution.

References desunt—The Editor

Text-based Knowledge Graph Embeddings

Andrea Maurino, Federico Bianchi, and Marco Cremaschi

INSID&S Lab
University of Milan-Bicocca
{firstname.lastname}@unimib.it

In the following we describe our project "Text-based Knowledge Graph Embeddings". We plan to use text to generate embeddings of entities that are describe in a Knowledge Graph. We want to extend our current work by using new text sources to generate the embeddings, considering embedding in hyperbolic spaces and introducing reasoning mechanisms inside our vector representations.

1 Introduction

Knowledge Graphs (KGs in the follow) have become a widespread abstraction model to represent knowledge about entities in a variety of domains, often based on graph databases. In a KG, nodes represent real-world entities and edges represent labeled relations between entities; a labeled relation can be seen as a triple (h, l, t) stating that the relation l holds between the entities h and t; e.g., (Barack_Obama, birthplace, Honolulu) represents the birthplace of Barack Obama. KGs have become of interest in the recent years, not only in the scientific community but also in the industry, where companies like Google and Microsoft have developed methods to build, store, use and explore KGs [7]. The most known example of open KG is DBpedia, which is based on the data deriving from Wikipedia and freely accessible online.[1]

1.1 Knowledge Graph Embeddings

KGs can be also seen as high dimensional and sparse data structures, which can be optimized to be better consumed by algorithms. A natural graph technique is to embed KG entities and relations into a vector space of lower dimension. This approach not only improves the tractability of certain tasks executed over the data structure, but it also supports the discovery of latent relationships in the data. Knowledge graph embedding capture latent component of a knowledge graph and are able to represent them in a lower dimensional space. Several recent works in the literature have been proposed for this problem by using different embedding methodologies [4, 9, 15, 18]. The core idea in these works is that the vectors expressing the elements of the triple (h, l, t) have a geometric property that represents their graph relation. Namely, the vector associated to entity h plus the vector associated to relationship l

[1] http://dbpedia.org (last accessed 2020-04-01).

should be a point close to the vector representation of entity t [4]. A different strategy proposed to reduce the dimensionality of KGs is tensor factorization [5, 11]. It is, in fact, possible to represent triples of a KG as coordinates of a tensor, i.e., a 3-way matrix, and then use matrix factorization techniques to obtain a more compact representation. The main method of evaluation for knowledge graph embeddings is link prediction [16] (i.e., predicting a missing head or a missing tail from a triple). Some model are now considering external information to obtain better performance in link prediction tasks, this information can come from textual corpora [14, 18] or logical rule [8, 17]. These representations are also useful in other contexts: in fact they can be used to tackle different problems like recommendation [20] or disambiguation and linking [19].

2 Current State of Research

Most of the aforementioned embedding models that have been applied to KGs like DBpedia are fed only with triples represented in a KG, without taking advantage of additional information contained in the KG or in other corpora. One alternative approach that we are exploring at UniMiB is to learn these embeddings from texts where entities are described or mentioned, once these texts are processed using NLP techniques.

Our current approach takes texts that are semantically annotated, i.e., texts where mentions of entities in the KG are recognized [2]. For example, from the text "Obama, former president of USA [...]" we obtain the annotated text "Obama<Barack_Obama>, former president of USA<United_States_of_America> [...]". Then we use word2vec, an off-the-shelf word embedding approach [10] on top of entities extracted from the texts. With this algorithm, vector representations are generated from relations among entities found in the text, i.e., two entities that appear close to each other proportionally often in texts will be closer in the vector space. Finally, by extracting entity types from the KG, e.g., Politician, City, etc., we are able to learn embeddings also for types and represent entities and their types as concatenation of these vectors. These type vectors have valuable properties and have been used to provide a novel way of computing the similarity based on distributional semantics [3].

In our framework, vector representations are based on co-occurrence of entities in texts, which help encode similarity between entities found in a corpus. We are now investigating methods to use vector concatenation to represent more aspects of entities in addition to corpus-based relatedness and type, e.g., the entity time. One major advantage of embeddings generated using this approach is that knowledge encoded in the vector representations can change with the corpus even if the KG does not change. In addition, we have extended the word2vec-based paradigm for analogical reasoning in such a way this form of reasoning can be performed over entities rather than over words (e.g., with Paris_France and Paris_Texas, vs "Paris"), and considering multiple criteria such as typification or time [2]. These forms of reasoning can be applied to exploration of knowledge, e.g., to find the equivalent

of Paris (capital of France) in Uzbekistan, to find the book from which the Blade Runner movie was derived, to find the equivalent of Cristiano Ronaldo in 1983.

Recently we have proposed a method to embed periods of time into a vector space by considering time descriptions [3] and a method to efficiently and effectively represent word meaning over time [6].

3 Next Steps

3.1 Enhanced Text-based Embedding

We want to replace word2vec and use more complex and sophisticated deep learning models that are are now available, like RNNs and LSTMs. For example, RNNs can help generate enriched and more accurate vector representations of KGs by considering the order and function of words in a phrase. As a consequence, we could apply our framework to other important problems like link prediction, link correction and entity clustering.

We plan to generate KG embeddings from larger corpora. For example, EventRegistry[2] is a corpus available in the EW-Shopp and EuBusinessGraph projects that consists of annotated news extracted daily from 300,000 sources. Generating KG embedding from annotated news can reveal relationships between entities that are not even represented in KGs. However, this would require a faster infrastructure to learn the embedding, which we do not have at the moment.

3.2 Hyperbolic Embedding

Another family of embedding model consider embeddings tree-like structures in a hyperbolic metric space. There exists increasing evidence that for hierarchical complex networks hyperbolic spaces are a suitable alternative. Negative curved spaces capture in a better way hierarchical structures, for example random hyperbolic naturally exhibit topological properties that characterized this kind of complex networks as power-law degree distribution and strong clustering coefficient.

There are still few works [12] on this topic and it is a recent trend that is showing good results in terms of representation and similarity computation for items.

We have already started working on the topic and we have generated embeddings for the DBpedia ontology. We are currently working on comparison with our recent work [3].

3.3 First Results on Reasoning

Finally, one thing we would like to add inside our embeddings is a reasoning system. We are currently working on the Logic Tensor Network [13] to add reasoning

[2] http://eventregistry.org/ (last accessed 2020-04-01).

capabilities to our system. The Logic Tensor Network takes in input first order fuzzy logic formulas and generate embeddings for functions, predicates and atoms of the language. We have published some first results [1] in which we show that LTNs seem to be a promising model to use for deductive reasoning.

References

[1] F. Bianchi and P. Hitzler. "On the Capabilities of Logic Tensor Networks for Deductive Reasoning". In: *AAAI Spring Symposium MAKE*. 2019.

[2] F. Bianchi and M. Palmonari. "Joint Learning of Entity and Type Embeddings for Analogical Reasoning with Entities." In: *NL4AI@AI*IA*. 2017.

[3] F. Bianchi, M. Soto, M. Palmonari, and V. Cutrona. "Type vector representations from text: An empirical analysis". In: *DL4KGS@ESWC*. 2018.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. "Translating embeddings for modeling multi-relational data". In: *NIPS*. 2013, pages 2787–2795.

[5] K.-W. Chang, S. W.-t. Yih, B. Yang, and C. Meek. "Typed Tensor Decomposition of Knowledge Bases for Relation Extraction". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pages 1568–1579. DOI: 10.3115/v1/D14-1165.

[6] V. Di Carlo, F. Bianchi, and M. Palmonari. "Training Temporal Word Embeddings with a Compass". In: *AAAI*. 2019. DOI: 10.1609/aaai.v33i01.33016326.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. "Knowledge vault: A web-scale approach to probabilistic knowledge fusion". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pages 601–610. DOI: 10.1145/2623330.2623623.

[8] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. "Jointly embedding knowledge graphs and logical rules". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pages 192–202. DOI: 10.18653/v1/D16-1019.

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. "Learning Entity and Relation Embeddings for Knowledge Graph Completion." In: *AAAI*. 2015, pages 2181–2187. DOI: 10.1609/aaai.v29i1.9491.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality". In: *NIPS*. 2013, pages 3111–3119.

[11] M. Nickel, V. Tresp, and H.-P. Kriegel. "A three-way model for collective learning on multi-relational data". In: *ICML*. 2011, pages 809–816.

[12] M. Nickel and D. Kiela. "Poincaré embeddings for learning hierarchical representations". In: *NIPS*. 2017, pages 6338–6347.

[13] L. Serafini and A. S. d. Garcez. "Learning and reasoning with logic tensor networks". In: *Conference of the Italian Association for Artificial Intelligence*. Springer. 2016, pages 334–348. DOI: 10.1145/3019612.3019642.

[14] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. "Representing text for joint embedding of text and knowledge bases". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pages 1499–1509. DOI: 10.18653/v1/D15-1174.

[15] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. "Complex embeddings for simple link prediction". In: *International Conference on Machine Learning*. 2016, pages 2071–2080.

[16] Q. Wang, Z. Mao, B. Wang, and L. Guo. "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pages 2724–2743. DOI: 10.1109/TKDE.2017.2754499.

[17] Q. Wang, B. Wang, L. Guo, et al. "Knowledge Base Completion Using Embeddings and Rules." In: *IJCAI*. 2015, pages 1859–1866.

[18] Z. Wang, J. Zhang, J. Feng, and Z. Chen. "Knowledge Graph Embedding by Translating on Hyperplanes." In: *AAAI*. 2014, pages 1112–1119.

[19] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. "Joint learning of the embedding of words and entities for named entity disambiguation". In: *arXiv preprint arXiv:1601.01343* (2016). DOI: 10.18653/v1/K16-1025.

[20] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. "Collaborative knowledge base embedding for recommender systems". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pages 353–362. DOI: 10.1145/2939672.2939673.

Scalable Batch and Real-time Analytics of Trajectory Data

Rim Moussa

Computer Science Eng. Department
National Engineering School -University of Carthage
{rim.moussa}@enicarthage.rnu.tn

A large volume of sensor networks and trajectories of mobile objects are collected. Such data offer us high value knowledge to understand moving objects and locations, fostering a broad range of applications in smart cities, enabling intelligent transportation systems and intelligent urban computing. Consequently, we need to engineer scalable and smart Trajectory Data Analytics Systems in order to analyze both historical data and real-time data flows.

1 Introduction

Trajectory data feature characteristics, which traditional systems cannot handle, such as (*i*) *high volume* — referring to the large quantity of generated and stored data; (*ii*) *high velocity* — referring to the speed at which new data is generated and processed; (*iii*) *high variety* — referring to multi-type and multi-source of data. Indeed, different types of data (structured, semi-structured and unstructured: raster data) are spotted for trajectory data; (*iv*) *High veracity* — referring to the high quality and value of captured and analyzed data.

Researchers and industrials promote intelligence in processing big trajectory data and development of connected smart transportation systems, in order to (*i*) increase of systems' autonomy, (*ii*) increase of safety, (*iii*) increase of productivity and (*iv*) reduction of congestion.

Contributions

Our work is mainly devoted to four important goals: (*i*) analysis of trips' patterns; (*ii*) analysis of impacts of other sources such as weather data, events on trips' patterns; (*iii*) investigation of advanced data structures in order to improve mining of trips' patterns; (*iv*) investigation of Lambda architectures for improving accuracy of RT and NRT queries.

2 Problem

For scalable analytics of trajectory data, we investigate the following functional requirements:

- *Support of Spatial On-Line Analytical Processing (SOLAP):* OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives. OLAP consists of four basic analytical operations: consolidation (roll-up), drill-down, and slicing (filtering along one dimension) and dicing (filtering along more than one dimension). The coupling of *Geographic Information Systems* (GIS) and *OLAP* has lead to the concept of *Spatial OLAP* (SOLAP) where GIS provides the cartographic representation and OLAP provides multidimensional perspective of data.
- *Support of Spatial Data Mining (SDM):* The coupling of *Geographic Information Systems* (GIS) and *Data Mining* (DM) has lead to the concept of *Spatial Data Mining* (SDM). The latter has emerged as a new area for spatial data analysis and as the process of discovering, interesting and useful non trivial patterns from spatial datasets using data mining algorithms. It is necessary to provide effective mining algorithms to extract knowledge from the trajectory data.
  Two types of algorithms are necessary,
  - Algorithms for learning trajectory patterns from historical data such as spatial outliers, spatial clusters for detecting hot-spots, Co-location patterns, e.g. weather conditions and trips patterns, Stay Points, trips' trajectory patterns, driving and speed patterns.
  - Algorithms for predicting future events such a vessel, cab or aircraft destination, future traffic congestion, trip's cost, et cetera.
- *Support of both batch and stream processing:* It's important to distinguish two types of processing models and underlying systems, namely *batch analytics* enabling retrospective analytics and *real-time analytics*. Batch systems allow processing of a large volume of historical data all at once, while real-time systems guarantee that the reaction will be within a tight real-world deadline, usually in a matter of seconds or milliseconds. Real-time systems software architecture is based on stream processing. Advanced architectures such as Lambda architecture [4] combine batch systems and real-time systems.

3 Solution

We propose scalable architecture and algorithms for scalable historical data exploration and near real-time prediction of trajectory data and congestion avoidance through calculation of future capacity of each geographical region.

4 Implementation

We are mainly using the following technologies Apache Spark, Neo4j, CAPS, GraphFrames and MLib. We are working on open data datasets related to (*i*) maritime data, released by the Danish Maritime Authority as well as (*ii*) ground transportation

and specifically cabs trajectory data, released by NYC Taxi and Limousine Commission (TLC).

4.1 Maritime Trajectory Data

The *Grand Challenge DEBS GC 2018* [2] deals with computing trips' patterns in order to predict *arrival-port* and *arrival-time* for vessels on sea. *MarineTraffic* provided an anonymized dataset which about 1 million of AIS tracks. Learning process is supervised. Indeed, the training dataset includes departure-port, ETA and destination-port. In [5] we propose scalable algorithms allowing *primo* to infer a map of vessels' trajectories and *secundo* to predict future locations of a vessel on sea.

4.1.1 Danish Maritime Authority AIS Data Preparation

The Danish Maritime Authority makes historical AIS data available (2 TB) [1], and 2 GB are daily generated. The data excerpt illustrated in listing 1 does not show any information about the departure-port and the arrival-port. The World Port Index (Pub 150) contains the location and physical characteristics and the facilities and services offered by major ports and terminals worldwide (3669 entries). The spatial data is in DMS (degrees, minutes, seconds) format (EPSG:4326). We succeed to:

- compute useful data from the World Port Index after converting into Lat-Lon format (WGS84 coordinate reference system) UTM data (EPSG:4326).
- perform the spatial join of the big historical dataset provided by the Danish Maritime Authority and the small dataset *World Port Index*.

4.2 Ground Trajectory Data

In [3, 6], we overview different geo-referenced data analytics systems such as Elastic Stack, GeoMondrian and Leaflet/Postgres. We succeed to:

- Build a NYC city subway graph database from a list of subway stations with visual checking of the correctness generated subway map.

Listing 1: Excerpt of AIS data released by the Danish Maritime Authority (Top 3 of ftp://ftp.ais.dk/ais_data/aisdk_20180501.csv).

```
Timestamp, Type of mobile, MMSI, Latitude, Longitude, Navigational status,
ROT, SOG, COG, Heading, IMO, Callsign, Name, Ship type, Cargo type,
Width, Length, Type of position fixing device, Draught, Destination, ETA,
Data source type

01/06/2018 00:00:00, Class A, 215810000, 55.921028, 17.320418, Constrained
by her draught, 0, 12.5, 255.5, 256, Unknown,,, Undefined,,,, Undefined,
,,, AIS
```

- Implementation of OLAP operations on top of two graph databases of NYC cabs trajectory data (2 GB), one hundred of the real dataset [7]. The latter is 200 GB.

4.3 Aircrafts Trajectory Data

We are assessing our contribution to *Aircraft Localization Competition* organized by *OpenSky Network*, Montreal, April 2019.[1]

5 Evaluation

We tested on desktop daily (2 GB) and monthly (20 GB compressed) maritime data from the Danish Maritime Authority. We also tested ground trajectory data based on excerpts from NYC yellow and green Cabs datasets.
Large scale experiments on Future SocLab are on-going.

6 Conclusion

Future work is mainly devoted to large scale experiments and publication of performances results.

References

[1] Danish Maritime Authority. *Danish Maritime Authority makes historical AIS data available*. 2018. URL: ftp://ftp.ais.dk/ais_data/ (last accessed 2018-08-30).

[2] V. Gulisano, Z. Jerzak, P. Smirnov, M. Strohbach, H. Ziekow, and D. Zissis. "The DEBS 2018 Grand Challenge". In: *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*. DEBS '18. Hamilton, New Zealand: Association for Computing Machinery, 2018, pages 191–194. ISBN: 978-1-4503-5782-1. DOI: 10.1145/3210284.3220510.

[3] A. Haddad. *Towards Scalable Analytics of Trajectory Data: Case of study New York City Cabs' Trips Warehouse*. Graduation Report in fulfillment of the requirements for the degree of Computer Science Engineer at ENI-Carthage. 2018.

[4] N. Marz and J. Warren. *Big Data Principles and best practices of scalable realtime data systems*. Manning Edition, 2015.

[1] https://competition.opensky-network.org/ (last accessed 2020-04-01).

[5] R. Moussa. "Scalable Maritime Traffic Map Inference and Real-Time Prediction of Vessels' Future Locations on Apache Spark". In: *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*. DEBS '18. Hamilton, New Zealand: Association for Computing Machinery, 2018, pages 213–216. ISBN: 978-1-4503-5782-1. DOI: 10.1145/3210284.3220506.

[6] R. Moussa, A. Haddad, and T. Bejaoui. "The Quest for Scalable and Intelligent Trajectory Data Analytics Systems: Status Report and Future Directions". In: *Proceedings of the 1st IEEE International Conference on SmartNets*. Hammamet, Tunisia, 2018.

[7] NYC Taxi and Limousine Commission. *TLC Trip Record Data*. 2018. URL: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml (last accessed 2018-03-08).

Machine Learning-Based Classification of Lung Diseases

Benjamin M. Abdel-Karim, Nicolas Pfeuffer, and Oliver Hinz

Information Systems and Information Management
Goethe University Frankfurt
abdel-karim,pfeuffer,hinz@wiwi.uni-frankfurt.de

The radiological evaluation of lung tissue is pivotal for an accurate diagnosis of lung diseases. Current advances in machine learning technologies have forwarded the automatic detection of pathological structures in human tissue based on radiographs. Current advances in machine learning technologies have the potential to increase the diagnosis precision and may help to reduce human errors. In this study we present a novel approach for the use of machine learning to improve the general decision process that relies on the judge-advisor theory and machine learning in a medical context. Our goal is to help doctors diagnose conditions so that they have more time for patients and their treatment. Our approach uses an artificial neuronal network to analyze the lung radiographs. Recent research has shown that this machine learning method is the state-of-the-art algorithm for image classification. We test our approach empirically.

1 Introduction

The radiological evaluation of lung tissue is pivotal for an accurate diagnosis of lung diseases. Current advances in machine learning technologies have forwarded the automatic detection of pathological structures in human tissue based on radiographs. Such automated image analysis approaches have great potential to increase the diagnosis precision and may help to reduce human errors. But unfortunately automated classification of lung diseases using radiographs is a challenging task owing to the fine-grained variability in the appearance of lung tissue. In this project we present a novel machine learning-based approach to improve the classification of lung diseases. Our approach employs a purpose-built artificial neuronal network (Convolution Neural Network (CNN)) to analyze lung radiographs.

The relevance of making a correct medical diagnosis cannot be overstressed [7]. Numerous studies have shown promising approaches. Accurate classification of cancer is a key basis for medicals and patient. Related research that focuses on cancer classification demonstrates the adequate performance of deep convolutional neural networks (see, for example, [3]). But the healthcare industry implements information technology relatively slowly [8]. This is surprising because the automation of sub-processes can significantly reduce the error rate [1].

In this study, we address the practical implications of the potential of CNN in the context of lung disease detection. In order to understand the potential of machine learning in lung disease detection we try to answer the following primary research

question (RQ): Is it possible to develop a machine learning approach to detect lung diseases reliable?

For this study, we have collected a huge dataset of more than 119 000 lung images from patients that suffer from one out of 15 different widespread lung diseases and which can be used to train the artificial neuronal network.

This study is organized as follows: First, we analyse previous research on three different research areas. We take a deeper look on convolutional neural network (CNN). In doing so, we also try to give an overview of our research methods. Here we describe an experiment to support expert knowledge by using CNN. We then turn to the analysis of our empirical study of improving our model in the result section. The last section offers the implications and a discussion of this study.

2 Previous Research

Convolutional neural networks (CNNs), which are assembled of multiple process layers to learn the representations of data with multiple abstract levels, are the most successful machine learning models in recent years [6]. CNNs are extensively used in image classification, obtaining encouraging classification accuracy over large-scale datasets compared to hand-engineered features based methods [10].

CNN is based on Artificial Neural Networks (ANNs) with multiple layers of neurons and is defined as a method of deep learning. The Convolutional Neural Network has grown in popularity by using image data and is now the state of the art for image classification. In a standard convolution network, a complete layer of a convolution network consists of three stages [6]. The CNN basic stages are the following: Convolution → Pooling → Fully Connected Layer → Output. The structure results in their essential phase in the CNN working.

The important stage is the convolution layer in which the convolution operation is executed to form a group of feature images. The second stage is the detection layer, in which any feature value is transmitted and activated to a nonlinear activation unit. The third stage is the pooling layer, where the features extracted on the lower layer are picked for sampling, which makes the network smaller [6]. The result is the output of the network.

Recent work includes an overview review on the future promise of interstitial disease detection and classification with deep learning are the works of [3] or [15]. Especially for lung diseases the work of [12] to be mentioned. The work of [2] shows the using of CNN in the case of cerebral microbleeds detection. [11] using CNN for automated pancreas segmentation or [13] using CNN as well for pulmonary nodule detection. In this case the work of [9] trained a single CNN to segment six tissues in MR brain images.

3 Data Description and Methods

For this study, first we use a special CNN as a state-of-the-art machine learning approach to classify pictures. We decided to use X-ray images, because they are one of the most commonly accessible radiological examinations for screening and diagnosis of many lung diseases [15]. Each patient first contacts his home doctor which usually starts with the analysis of an X-ray image. In this study, we use a new chest X-ray database, namely "ChestX-ray8", which comprises 112,100 frontal-view X-ray images of 32,717 unique patients with the text-mined eight disease image labels (where each image can have multi-labels) for the National Institutes of Health (NIH) in the United States of America. We use an open-access dataset repository from the NIH clinical center - Americans research hospital.[1] For the sample selection we have decided to focus on the eight most common lung diseases [15].

In the case of data preparation we decided to remove blurred and unclear images from the test and validation data set. These are helpful training data, but great care has been taken to ensure that these sets are not split between the training and validation sets. Although the network accepts image inputs of 299 × 299 pixels. Therefore we reduce the image size for our machine learning approach to a size of 299 × 299 pixels. To standardize this process we have written our own algorithm to resize the x-ray images. However, the diseases are distributed differently in the dataset. For this reason, we have extended the individual classes accordingly, as shown for example in the work of [3], so that the number of images is evenly distributed with the diseases. Classes with inclined images are determined by the random selection of existing images and this selected image is rotated randomly between $-10°$ and $10°$. HPI provided us access to 2 GPUs and we use therefore two Tesla K80 Cards for calculation.

4 Results and next Steps

In our setting we use initial weights from the ImageNet dataset; therefore we follow the state-of-the-art concept of transfer-learning for ANN in line with works of [3, 5]. This means that the model is pre-trained on 1,000 object classes (1.28 million images) of the 2014 ImageNet Challenge [14]. Our training methodology is based on the pre-trained Inception V3 architecture. In the context of fine tuning we use as starting point the parameters that recent literature points to. Therefore all layers use the same global learning rate of 0.001 and a decay factor of 16 [14]. Additionally, we use the RMSProb with a decay of 0.9, momentum of 0.9 and elision of 0.10 [3]. We decided to use 3000 epochs to learn this model in consequence of the result of the current research [4]. Our results at this point of time indicate several problems. The figure shows an example of the primary problem.

[1] https://nihcc.app.box.com/v/ChestXray-NIHCC (last accessed 2020-04-01).

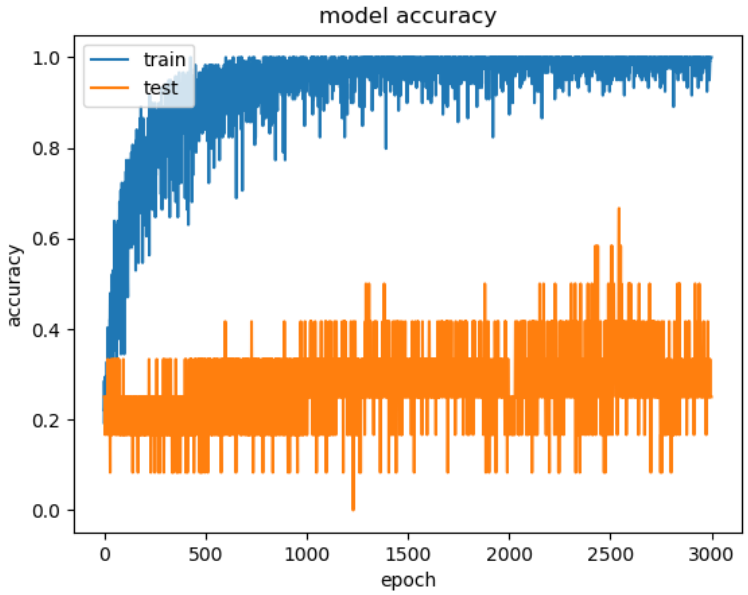


Figure 1: Train Process

The network memorizes the images in the test set. However, the test images for the network are hardly recognizable. After systematic testing of different parameter combinations and modifications of pre-processing techniques we decided to change the network architecture.

References

[1] R. Aron, S. Dutta, R. Janakiraman, and P. A. Pathak. "The Impact of Automation of Systems on Medical Errors: Evidence from Field Research". In: *Information Systems Research* 22.3 (2011), pages 429–446. DOI: 10.1287/isre.1110.0350.

[2] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, M. VC, L. Shi, and P.-A. Heng. "Automatic Detection of Cerebral Microbleeds From MR Images via 3D Convolutional Neural Networks". In: *IEEE Transactions on Medical Imaging* 35.2 (2016), pages 1182–1195. DOI: 10.1109/TMI.2016.2528129.

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.2 (2017), pages 115–127. DOI: 10.1038/nature21056.

[4] P. Khosravi, E. Kazemi, M. Imielinski, O. Elemento, and I. Hajirasouliha. "Deep Convolutional Neural Networks Enable Discrimination of Heterogeneous Digital Pathology Images". In: *EBioMedicine* 27.1 (2018), pages 317–328. DOI: 10.1016/j.ebiom.2017.12.026.

[5] A. Kurakin, I. J. Goodfellow, and S. Bengio. "Adversarial Machine Learning at Scale". In: *International Conference on Learning Representations* 5 (2017), pages 1–17.

[6] S.-J. Lee, T. Chen, L. Yu, and C.-H. Lai. "Image Classification Based on the Boost Convolutional Neural Network". In: *IEEE Access* 6.3 (2018), pages 12755–12768. DOI: 10.1109/ACCESS.2018.2796722.

[7] P. Mangiameli, D. West, and R. Rampal. "Model selection for medical diagnosis decision support systems". In: *Decision Support Systems* 36.3 (2004), pages 247–259. DOI: 10.1016/S0167-9236(02)00143-4.

[8] N. M. Menon, B. Lee, and L. Eldenburg. "Productivity of Information Systems in the Healthcare Industry". In: *Information Systems Research* 1.3 (2000), pages 83–92. DOI: 10.1287/isre.11.1.83.11784.

[9] P. Moeskops, J. M. Wolterink, B. H. van der Velden, K. G. Gilhuijs, T. Leiner, M. A. Viergever, and I. Išgum. "Deep learning for multi-task medical image segmentation in multiple modalities". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI* (2016), pages 478–486. DOI: 10.1007/978-3-319-46723-8_55.

[10] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik. "Convolutional Neural Networks Based Fire Detection in Surveillance Videos". In: *IEEE Access* 6.3 (2018), pages 18174–18182. DOI: 10.1109/ACCESS.2018.2812835.

[11] H. R. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu1, E. Turkbey, and R. M. Summers. "DeepOrgan: Multi-level Deep Convolutional Networks for Automated Pancreas Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015* (2015), pages 556–564. DOI: 10.1007/978-3-319-24553-9_68.

[12] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers. "A New 2.5D Representation for Lymph Node Detection using Random Sets of Deep Convolutional Neural Network Observations". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI* 8673.1 (2014), pages 520–527. DOI: 10.1007/978-3-319-10404-1_65.

[13] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken. "Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pages 1160–1169. DOI: 10.1109/TMI.2016.2536809.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *Computer Vision and Pattern Recognition* 11.12 (2015), pages 1–10. DOI: 10.1109/CVPR.2016.308.

[15] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". In: *IEEE CVPR 2017* 27.9 (2017), pages 2097–2106. DOI: 10.1109/CVPR.2017.369.

Integrating Hardware Accelerators in Virtualized Environments

A Preliminary Evaluation of I/O Link Compression in Heterogeneous Systems

Max Plauth and Andreas Polze

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

The overhead of moving data is the major limiting factor in todays hardware, especially in heterogeneous systems where data needs to be transferred back and forth frequently between host and accelerator memory. In the *Fall 2018* period, this project investigated the potential of *On-the-Fly I/O Link Compression* as a promising approach to reduce data volumes and transfer time, thus improving the overall efficiency of accelerators in heterogeneous systems.

1 Introduction

With the increasing prevalence of big data applications, data volumes are growing by the day and compute tasks are gaining complexity. Even though hardware accelerators such as GPUs or Field-Programmable Gate Arrays (FPGAs) are vital for satisfying these demands, moving data back and forth between main memory and accelerators remains a major limiting factor, which is reflected by the many consortia working on faster interconnection technologies (e.g. OpenCAPI [14], CCIX [1], or Gen-Z [9]).

Contributions

Even though several strategies exist for mitigating the performance impact of memory transfers (e.g. by overlapping computation and data transfers), these mitigation strategies are only applicable to workloads that can be pipelined. Preceding efforts of the research community have identified compression as an viable orthogonal method for improving data transfer efficiency for many application domains [6]. In an attempt to anticipate hardware-accelerated compression becoming increasingly available in many computer architectures [3, 11], we used a software-based implementation of the 842 compression algorithm [5, 17] to evaluate the benefits of using *On-the-Fly I/O Link Compression* to reduce data volumes and thus data transfer time in heterogeneous systems.

2 Background: 842 Compression

The 842 compression algorithm has been introduced by IBM and has been implemented in hardware in the *nx842* on-chip compression accelerator available in their POWER processors.The main design goal of the 842 algorithm [5] is to allow high-throughput/low-latency hardware implementations that can be placed directly on transmission channels [17]. These properties do align perfectly with our intent of evaluating the feasibility of hardware-accelerated *On-the-Fly I/O Link Compression*. Hereinafter, the basic procedure of the 842 algorithm is outlined.

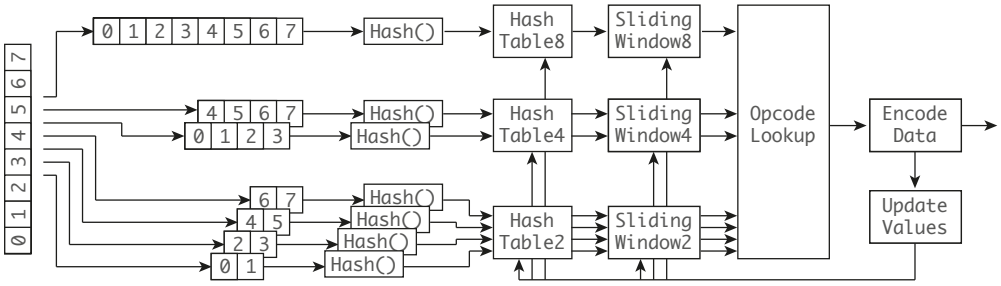


Figure 1: The 842 compression algorithm operates on units of 8 bytes, treating the input data as sub-phrases of 8, 4 and 2 bytes length. The algorithm relies on efficient hashing and sliding window buffers containing past compressed data.

As illustrated in figure 1, the 842 algorithm [5] implemented by the *nx842* on-chip accelerator operates on units of 8 bytes, treating the input data as sub-phrases of 8, 4 and 2 bytes length, respectively. For each phrase length, a hash function and a hash table with offsets to a sliding window buffer of past encoded data are used to detect possible matches of the sub-phrase therein. Based on the lookup, a template is chosen that encodes 8 bytes of raw data. Each 5-bit template encodes a permutation of offsets or literals of 8, 4 and 2 bytes length, followed the actual offsets and literals. With a clock frequency of 2.3 GHz, and the ability to ingest 8 bytes per cycle, one *nx842* on-chip accelerator can achieve a maximum throughput of 18 GB/s [11]. Being equipped with two *nx842* on-chip accelerators [2], the total compression throughput of a POWER8 processor can be as high as 36 GB/s.

The 842 algorithm can be attributed to the family of Lempel-Ziv derivatives [5]. The compression process deviates from the original Lempel-Ziv algorithm [18] in several aspects. However, decompression works almost identical compared to LZ'77 [5]. With a sufficient number of approaches available that have demonstrated efficient decompression of Lempel-Ziv derivatives on GPUs [7, 8, 15, 16, 19], we assume that decompression of 842-encoded data can be implemented efficiently on GPUs. Regarding FPGAs, an efficient implementation of 842 for both compression and decompression has been demonstrated as well [17], providing further indication that efficient implementations of the algorithm are feasible for most popular accelerator types.

3 Experimental Setup

While *link speed* is a constant determined by the deployment environment, *compression throughput* and the *space savings* for a given workload are the decisive variables deciding whether *On-the-Fly I/O Link Compression* is feasible or not. Hereinafter, this section elucidates the experimental procedures used to obtain practical values for both variables.

3.1 Space Savings

In order to ascertain the space savings achieved by the 842 algorithm for a wide set of versatile workloads, we compiled a well-defined suite of test data. The test data suite is comprised of the *Large Text Compression Benchmark* [13], the Silesia Corpus [4], the first chromosome of the *GRCh38.p12 Human Reference Genome* [10], two grayscale images (4240 × 2832 pixels, TIFF file), as well as a rasterized image of figure 1 (3206 × 910 pixels, TIFF file).

3.2 Compression Throughput

In our evaluation, we included a number of different software-based implementations of the 842 algorithm. The naïve software implementation available in the Linux kernel has been used as a baseline (*SW/K: Linux*). Additionally, the kernel-based implementation has been extracted as a user space application (*SW/U: Naïve*), serving as the foundation for an optimized software implementation (*SW/U: Optimized*).

All measurements have been performed on the test system specified in table 1. All test procedures and the software-based implementations are freely available on GitHub [12].

Table 1: Specifications of bare-metal test environment

	IBM Power System S824L
CPU	2 × IBM POWER8, 10 Cores / 80 Threads, 3.42 GHz
Memory	16 × 64 GB DDR3 ECC CDIMM, 1600 MHz
GPU	1x NVIDIA Tesla K80
OS	Ubuntu 16.04.4 64 Bit

4 Results

This section presents practical measurements of space savings and compression throughput as suggested in section 3. The *space savings* yielded by the 842 algorithm for the test data set are reported in figure 2.

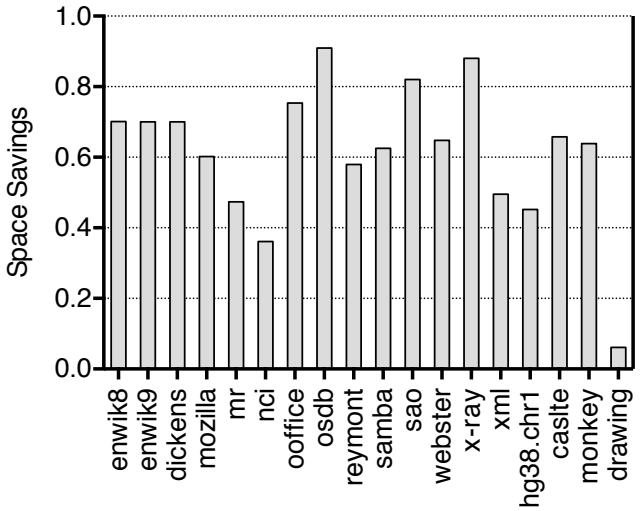


Figure 2: Space savings achieved by the 842 algorithm for the test data suite

The results of the *compression throughput* measurements are presented in figure 3.

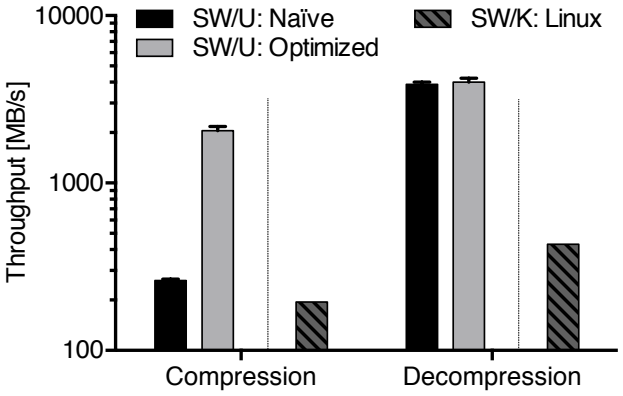


Figure 3: (De-) Compression throughput measured for soft- and hardware-based implementations of the 842 algorithm

5 Outlook

To study the feasibility of *On-the-Fly I/O Link Compression* in practice, the next logical steps include completing the test setup elaborated in this paper with accelerator-based decompression facilities to cover the full *compress-send-decompress* workflow. Furthermore, to increase the expressiveness of our investigation, additional experiments need to be conducted using benchmarks from the heterogeneous computing domain instead of benchmarks from the data compression domain.

References

[1] *CCIX Consortium*. URL: https://www.ccixconsortium.com (last accessed 2019-03-08).

[2] S. Chabrolles, J. Limodin, and F. Moyen. *Enabling POWER 8 Advanced Features on Linux*. 2016.

[3] *Datapath Acceleration Architecture 2*. NXP Semiconductors. Mar. 8, 2019.

[4] S. Deorowicz. *Silesia Compression Corpus*. Silesian University of Technology, Poland. 2003. URL: http://sun.aei.polsl.pl/~sdeor/ (last accessed 2019-03-08).

[5] P. A. Franaszek, L. A. Lastras-Montaño, S. Peng, and J. T. Robinson. "Data Compression with Restricted Parsings". In: *Data Compression Conference (DCC'06)*. Mar. 2006, pages 203–212. DOI: 10.1109/DCC.2006.22.

[6] S. Funasaka, K. Nakano, and Y. Ito. "Adaptive Loss-Less Data Compression Method Optimized for GPU Decompression". In: *Concurrency and Computation: Practice and Experience* 29.24 (Dec. 2017), e4283. DOI: 10.1002/cpe.4283.

[7] S. Funasaka, K. Nakano, and Y. Ito. "Fast LZW Compression Using a GPU". In: *2015 Third International Symposium on Computing and Networking (CANDAR)*. Dec. 2015, pages 303–308. DOI: 10.1109/CANDAR.2015.20.

[8] S. Funasaka, K. Nakano, and Y. Ito. "Fully Parallelized LZW Decompression for CUDA-Enabled GPUs". In: *IEICE Transactions on Information and Systems* 99.12 (2016), pages 2986–2994.

[9] *Gen-Z Consortium*. Memento. Orignal defunct. Jan. 1, 2019. URL: https://web.archive.org/web/20190111214524/https://genzconsortium.org/ (last accessed 2019-03-08).

[10] *GRCh38.p12 Human Reference Genome*. Genome Reference Consortium, Dec. 2017. URL: https://www.ncbi.nlm.nih.gov/grc/human.

[11] H. Hellmuth and J. Klauke. *POWER NX842 Compression for Db2*. White Paper. IBM Corporation, Sept. 2017.

[12] *Lib842 Repository*. URL: https://github.com/plauth/lib842 (last accessed 2019-03-08).

[13] M. Mahoney. *Large Text Compression Benchmark*. URL: http://mattmahoney.net/dc/textdata.html (last accessed 2019-03-08).

[14] *OpenCAPI Consortium*. URL: https://opencapi.org (last accessed 2019-03-08).

[15] A. Ozsoy and M. Swany. "CULZSS: LZSS Lossless Data Compression on CUDA". In: *2011 IEEE International Conference on Cluster Computing*. Sept. 2011, pages 403–411. DOI: 10.1109/CLUSTER.2011.52.

[16] E. Sitaridi, R. Mueller, T. Kaldewey, G. Lohman, and K. A. Ross. "Massively-Parallel Lossless Data Decompression". In: *2016 45th International Conference on Parallel Processing (ICPP)*. Aug. 2016, pages 242–247. DOI: 10.1109/ICPP.2016.35.

[17] S. Suresh and V. Udayashekar. "High-Throughput, Lossless Data Compression and Decompression On FPGAs". Master's Thesis. California State University, Northridge, May 2012, page 84.

[18] J. Ziv and A. Lempel. "A universal algorithm for sequential data compression". In: *IEEE Transactions on Information Theory* 23.3 (May 1977), pages 337–343. ISSN: 0018-9448. DOI: 10.1109/TIT.1977.1055714.

[19] Y. Zu and B. Hua. "GLZSS: LZSS Lossless Data Compression Can Be Faster". In: *Proceedings of Workshop on General Purpose Processing Using GPUs*. Salt Lake City, UT, USA: ACM, 2014, 46:46–46:53. ISBN: 978-1-4503-2766-4. DOI: 10.1145/2588768.2576785.

Expanding Semantic Tag-Based Representation Learning

Da Huo and Gerard de Melo

Deep Data Lab
Rutgers University, New Brunswick, NJ, USA
dh637@scarletmail.rutgers.edu, gdm@demelo.org

Vector representations are widely used in machine learning and can enable us to impart background information about items into a learning process. In this work, we consider word vector representations that convey explicit and interpretable information about the semantic properties of words, drawing on the recently proposed notion of semantic tagging. We study to what extent we can automatically extend the coverage of our initial word vector representations to flexibly cover new words that do not occur in the original corpus. Our experiments show that this succeeds with high accuracy, even across languages.

1 Introduction

Motivation In machine learning, vector representations are now widely appreciated for their ability to impart background information about items into a learning process. For example, the machine learning algorithm may never have seen the string *Havel* in its training data, but if the representation for *Havel* reveals that it denotes a European river similar to others such as *Elbe* and *Danube*, the machine learning algorithm may be able to treat it appropriately. While there are publicly available word vectors capturing the general statistical co-occurrrence-based similarity of different words [11], in this work, we consider representations based on more explicit and interpretable semantic properties of words. Traditional part-of-speech tagging has focused on distinctions based on the grammatical function of words. In contrast, the recently proposed task of *Semantic Tagging* [4] considers a set of tags that are conjectured to be pertinent in semantic tasks. The annotation scheme distinguishes, for instance, privative attributes (`PRI`) such as *former* or *fake* from intersective ones (`IST`) such as *vegetarian*. Abzianidze and Bos [3] presented an improved tag set and showed that the tags exhibit the potential to apply cross-lingually. The Parallel Meaning Bank project [2] provides a parallel corpus that includes semantic tag annotations. However, just up to a few thousand sentences have been annotated per language.

Project Idea In this paper, we study to what extent automatic means can be used to expand the coverage of Semantic Tag-based word vector representations. The goal is to dynamically infer similar kinds of representations for new words or tokens that did not occur in the annotated corpus. As input, we rely on a list of words along with their regular semantic word vector embeddings.

We first induce interpretable semantic tag vector representations from the annotated corpus. We then predict such vector representations for new words based on related words from the generic large-scale word representation data.

2 Method

2.1 Initial Semantic Tag-based Vector Representations

Input As input, we assume an annotated corpus $\mathbb{C} = ((w_1, t_1), \ldots, (w_L, t_L))$, in which word tokens w_i have been annotated with tags $t_i \in \mathbb{T}$ from a global tag set $\mathbb{T}$. Here, the vocabulary of tokens $\mathbb{V} = \bigcup_{w_i \in \mathbb{C}} \{w_i\}$ contains consists of (string,part-of-speech) tuples.

Seed Tag Vectors Given this data, we first compute seed vectors $\mathbf{v}_w$ for $w \in \mathbb{V}$ as $\mathbf{v}_w = \sum_{(w_i,t_i) \in \mathbb{C}: w_i = w} \mathbf{e}(t_i)$. Here, $\mathbf{e}(t)$ is a function yielding a $|\mathbb{T}|$-dimensional vector with a one-hot encoding of tag t. The resulting semantic tag vectors hence capture the distribution of a word's tags.

2.2 Dynamic Extension of Vector Representations

Having computed such tag vector representations for each word observed in our annotated corpus $\mathbb{C}$, we now wish to infer similar kinds of semantic tag vectors $\mathbf{v}_w$ for unseen input words w. We proceed in two steps. First retrieve relevant neighbors of every such new unseen word. Then, we use those neighbors to induce a vector for the unseen word.

Neighbor Retrieval Given a target unseen word w, we first determine its k nearest neighbors $N_k(w)$ using a preexisting word embedding matrix E as $N_k(w) = \sigma_k(w, E, \mathbb{V}_E \cap \mathbb{V})$. Here, σ_k is a function yielding a set of the k closest words in the embedding space E with respect to a vocabulary. It retrieves the regular word vector $\mathbf{u}_w$ for the input word w in the word embedding space E, and computes its k nearest neighbors in E in terms of Euclidean distance. However, for this, σ_k only considers the subset of words that are in $\mathbb{V}_E \cap \mathbb{V}$, i.e., those words that are in the vocabulary $\mathbb{V}_E$ of E, but also in the annotated corpus vocabulary $\mathbb{V}$. We assume an E providing embeddings for specific lemma and part-of-speech combinations of words, i.e., σ_k retains only those vocabulary items with a compatible part-of-speech tag to w, while still seeking to maintain that $|N_k(w)| = k$.

Tag Vector Induction Finally, we predict semantic tag vectors $\mathbf{v}_w$ as

$$\mathbf{v}_w = \frac{1}{|N_k(w)|} \sum_{w' \in N_k(w)} \frac{\mathbf{u}_w \, \mathbf{u}_{w'}}{\|\mathbf{u}_w\| \, \|\mathbf{u}_{w'}\|} \, \mathbf{v}_{w'}. \quad (1)$$

Here, the various $\mathbf{v}_{w'}$ are semantic tag vectors for the k nearest neighbors in N_k.

Cross-Lingual Induction Our approach can also be extended for cross-lingual semantic tag vector induction. We use the same vector computation as above but need to take one additional step. Given a non-English word w_0, we first retrieve a set of English translations $W = \{w \mid (w_0, w) \in T, w \in \mathcal{V}_E\}$. from a translation dictionary T, which we assume provides part-of-speech specific translations. In other words, during this process, we only consider translations w that occur in the vocabulary of our English embeddings E with a matching part-of-speech tag. If $|W| > 0$, we can then obtain

$$\mathbf{v}_{w_0} = \frac{1}{|W|} \sum_{w \in W} \mathbf{v}_w, \tag{2}$$

where the $\mathbf{v}_w$ are computed using the monolingual method from Eq. 1.

3 Experimental Findings

Computational Resources Up to this point, this work has mostly been conducted on our own small-scale machines, because of the initial exploratory nature of devising new methods. HPI FSOC has provided access to an 8 core Intel Xeon E7-4870 2.40 GHz server, which we envision using to scale our process up to much larger data.

Data We rely on version 2.1.0 of the Parallel Meaning Bank (PMB) corpus [2], of which we consider the gold quality subset. As word embeddings E, we rely on the Sketch Engine English (Web, 2013, 20 billion tokens) Lemma + Part of Speech word embeddings.[1] It has a vocabulary size of 6,143,073. For our cross-lingual evaluations, we rely on translations extracted from the 2018-11-20 English edition of Wiktionary to generate English translations of the input word.

Experimental Protocol We tested our method on English as well as cross-lingually using gold data from the PMB corpus. For English words, besides the 90-dimensional semantic tag vector prediction experiments, we also conducted experiments to predict the more general 15-dimensional coarse-grained tags provided by the corpus (denoted as *English* (C)). Due to the small size of the seed data, we rely on a leave-one-out evaluation for each setting, i.e., we consider the gold data as the ground truth and try to predict each word's ground truth tag vector separately, based only on other seeds in the data, excluding the target word itself. We use the *average cosine similarity* score between the ground truth vector and the predicted vector to quantify the accuracy of our method.

Overall Results The experimental results are given in Table 1. Our approach obtains fairly high cosine similarities. Due to the use of part-of-speech-specific embeddings, the fine-grained English prediction with 90 tags is about as accurate as the

[1] https://embeddings.sketchengine.co.uk/ (last accessed 2020-04-01).

Table 1: Prediction results for different k

Language	1	2	3	4	5	10	20
English	0.76	0.77	0.78	0.78	0.78	0.77	0.78
English (C)	0.75	0.76	0.76	0.76	0.77	0.77	0.78
German	0.84	0.84	0.83	0.83	0.83	0.84	0.84
Dutch	0.86	0.84	0.84	0.85	0.84	0.84	0.83
Italian	0.82	0.84	0.82	0.83	0.83	0.86	0.86

coarse-grained prediction. The results are also fairly strong on cross-lingual mappings, despite the fact that this involves relying on a translation dictionary, which may bring in additional ambiguity and may result in a skewed tag distribution if the set of available translations is skewed towards particular word senses.

4 Conclusion and Next Steps

Vector representations based on Universal Semantic Tags are a promising new way of capturing salient semantic properties of words. Our research provides a method to dynamically extend the coverage of the original Universal Semantic Tag data to a large set of new word forms both in English and across many other languages. Abdou, Kulmizev, Ravishankar, Abzianidze, and Bos [1] demonstrated the usefulness of such semantic tags in several downstream tasks via multi-task learning, including on the Stanford NLI corpus, SICK, POS tagging, and dependency tagging. Hence, we envision our data being useful in a wide range of tasks that benefit from semantic information about words.

In terms of future work, the next phase of our research will involve scaling up our approach to account for information from a larger number of heterogeneous data sources [5, 6, 8], including structured data sources [9, 10]. This will enable us to provide accurate vector representations for a much larger range of items from numerous different languages. Another goal is to investigate the zero-shot learning and few-shot learning cases by better exploiting contextual features of words [7] as well as the internal morphology [12].

References

[1] M. Abdou, A. Kulmizev, V. Ravishankar, L. Abzianidze, and J. Bos. "What can we learn from Semantic Tagging?" In: *Proceedings of EMNLP 2018*. Oct. 2018, pages 4881–4889. DOI: 10.18653/v1/D18-1526. URL: https://aclweb.org/anthology/D18-1526.

[2] L. Abzianidze, J. Bjerva, K. Evang, H. Haagsma, R. van Noord, P. Ludmann, D.-D. Nguyen, and J. Bos. "The Parallel Meaning Bank: Towards a Multilingual

Corpus of Translations Annotated with Compositional Meaning Representations". In: *Proceedings of EACL 2017*. Valencia, Spain, Apr. 2017, pages 242–247. URL: https://aclweb.org/anthology/E17-2039.

[3] L. Abzianidze and J. Bos. "Towards Universal Semantic Tagging". In: *IWCS 2017— 12th International Conference on Computational Semantics*. 2017. URL: https://aclweb.org/anthology/W17-6901.

[4] J. Bjerva, B. Plank, and J. Bos. "Semantic Tagging with Deep Residual Networks". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, 2016, pages 3531–3541. URL: https://aclweb.org/anthology/C16-1333.

[5] J. Chen, N. Tandon, C. D. Hariman, and G. de Melo. "WebBrain: Joint Neural Learning of Large-Scale Commonsense Knowledge". In: *Proceedings of ISWC 2016*. Kobe, Japan, 2016.

[6] G. de Melo. "Wiktionary-Based Word Embeddings". In: *Proceedings of MT Summit XV*. Miami, FL, USA, 2015.

[7] C. Li, G. Wang, and G. de Melo. "Context-Based Few-Shot Word Representation Learning". In: *Proceedings of the 12th IEEE International Conference on Semantic Computing*. Laguna Hills, CA, USA, 2018.

[8] G. de Melo. "Inducing Conceptual Embedding Spaces from Wikipedia". In: *Proceedings of WWW 2017*. Perth, Australia: ACM, 2017.

[9] G. de Melo and G. Weikum. "Taxonomic Data Integration from Multilingual Wikipedia Editions". In: *Knowledge and Information Systems* 39.1 (Apr. 2014), pages 1–39. ISSN: 0219-1377. DOI: 10.1007/s10115-012-0597-3. URL: 10.1007/s10115-012-0597-3.

[10] G. de Melo and G. Weikum. "Towards a Universal Wordnet by Learning from Combined Evidence". In: *Proceedings of CIKM 2009*. ACM, 2009, pages 513–522. ISBN: 978-1-60558-512-3. DOI: 10.1145/1645953.1646020.

[11] J. Pennington, R. Socher, and C. D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pages 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclweb.org/anthology/D14-1162.

[12] L. Wang, Z. Cao, Y. Xia, and G. de Melo. "Morphological Segmentation with Window LSTM Neural Networks". In: *Proceedings of AAAI 2016*. Phoenix, AZ, USA: AAAI Press, 2016, pages 2842–2848.

Hybrid Modelling of Aluminium Smelting Bath Chemistry Process

An approach combining SAP HANA in-memory processing, machine learning algorithms and mathematical theoretical models

Fábio M. Soares and Roberto C. L. De Oliveira

Post-Graduation Program in Electrical Engineering
Federal University of Pará, Brazil
fms@ufpa.br, limao@ufpa.br

Aluminium smelting processes are typically complex and multidisciplinary, hard to be modeled and controlled. Smelters typically have hundreds of production cells, where each of them has a particular operation, i.e. it is not easy to design a general control strategy without taking these differences into account. Recent advances in data science have helped this industry to overcome some of these challenges by providing rich information, however good hardware capable of processing a great amount of data are often available on the cloud, whilst most of industries' plant floor have limited hardware and internet connectivity. It is therefore desirable that these hardware can be enabled to provide such useful information on the process, and that is where process modeling comes in. The high computing cloud based hardware is capable of processing the data and build simpler models, which are downloaded to plant floor hardware. Models are built using both mathematical and data-driven approaches. Considering the differences in production cells, in this report we show two data-driven tasks applied to the modelling of bath chemistry process variables, whereby we simulate some situations to check how accurate the forecasts are in the n-step ahead predictions.

1 Project Idea

Aluminium smelting involves many fields within chemistry and physics, and in a higher level, economics, i.e. it is multidisciplinary [4]. This is reflected in the huge databases produced by the smelters, whereby the management is nearly impossible without automation. However to achieve the maximum level of productivity still requires a lot of thinking and time-consuming tasks, such as the design of control strategies. Process modelling helps in planning productivity by forecasting trends and simulating alternative scenarios. Most of the modelling being made on aluminium smelting is based on the physical laws of the process[1, 2], which are too complex. Data science techniques, on the other hand enable a shortcut to build a model based on the process dynamics itself [3, 5]. This project combines mathematical models with data-driven algorithms as these approaches complement each other. Moreover

the data-driven models takes into account the differences among production cells, whereby they are divided into clusters that share the same behaviour throughout its operation time. This approach leads to the modelling of more accurate models, as cell operation history strongly dictates its dynamics.

1.1 Approach used

In the last periods' report, we have applied clustering and principal components analysis to model aluminium reduction cells to reduce the dataset both in dimensionality and number of samples. One of the greatest challenges faced so far was the accuracy of the models in the long term simulation, which still remains. Factors like noisy and poorly measured data explain this difficulty, however the mathematical models already help in proving a theoretical view on how the cell variables should behave under certain conditions. In addition, the cluster-based models show a significantly better performance in comparison to the approach of using reduction-based or smelter-based modelling. Regarding database completeness, many variables are already imputed, i.e. null values are produced by mathematical models, therefore we've managed to reduce the sampling time from days to hours. This means we have more data than the originally available plant floor database. We performed a new study on clustering aluminium reduction cells taking into account this updated database to find new cells' cluster. Then for each cluster we apply supervised learning algorithms, including linear regression and neural networks for cell modelling. The linear regression helps in modeling the linear behaviour of the cell and the neural network learns the non-linear behaviour. Figure 1 shows a scheme of the approach used in this stage of the project.

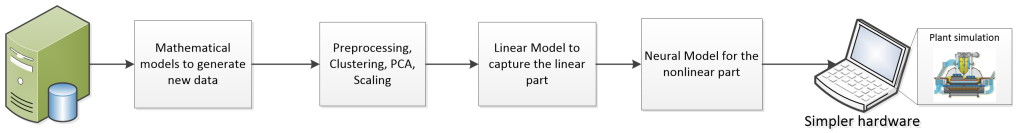


Figure 1: Approach used

1.2 Clustering and Preprocessing

Lima et al. [6] performed a study on clustering aluminium reduction cells using statistical measures such as mean, median and standard deviation, as suggested in the work of Horvath & Vircikova[5]. We used the same approach to find clusters of cells whose data will be applied to build models for each cluster separately, assuming all the cells in the cluster share close behaviours. We've also applied principal components analysis (PCA) to eliminate eventual redundancy in the data. In this report we compare the models we already have with the models obtained using data from clusters and using PCA.

2 Used FSOC resources

Our project used one instance of SAP HANA database server to store the data in a column-stored table, enabling fast aggregate operations, such as mean and sum. Also we loaded our models into 8 Virtual Machines; 4 of them are extra large to process the cluster models for each single cell from each of the 4 potlines, and the other 4 are only large to process and potline related data using the general models. The mathematical models that generate lump paramater data run on both extra large and large virtual machines, while the distributed parameter mathematical models, which generate spatial and time data for each variable, are run on the multicore server and the NVidia Tesla card. However the boundary conditions to generate these data couldn't be determined, as this depends also on measurements to be taken from the potlines.

3 Findings

In the previous periods we have built simple models with the least possible number of hidden neurons in multilayer perceptron regressors. We have found these models to predict accurately in the one-step ahead, but in the long-run (multi-step) the predictions need improvements. Retraining the models with updated data helped in improving performance, however as the database inputs are many, the retraining time increases. In this context we think of linear models that learn the linear behaviour of the cell, and leaves the neural network (NN) to focus on the nonlinear part [8]. Table 1 shows the results for two of the most bath chemistry variables already modeled comparing the test mean squared error (MSE) performed by the last clusted-based model vs. updated cluster-based models with linear+NN.

Table 1: MSE Errors comparing cluster based NN and clusted based linear+NN

	General based model		Cluster based model	
Steps	Liquidus Temp.	Fluoride Concentration	Liquidus Temp.	Fluoride Concentration
1	0.005 89	0.033 06	0.006 346	0.002 468
2	0.028 627 5	0.105 918	0.030 56	0.006 092 9
3	0.065 151 9	0.250 966	0.016 545	0.009 609 7

It is worth noting that the MSE is calculated over the normalized output value, subtracting the mean and diving by the standard deviation. The predictions get worse after the 2nd step (8 hours ahead).

3.1 Simulation

To verify the model's reliability we launched the model in parallel with a bath chemistry fuzzy control algorithm [7] to test the new version of the models, including retraining. We wanted to compare bath temperature (TMP) and aluminium fluoride (ALF), which are strongly related to each other. Both are controlled by adding aluminium fluoride to the cell (ALF3A). For this case, using PCA and cluster-based information, the results are for one-step prediction. If the error is more than 10 percent, the model is retrained with updated real data.

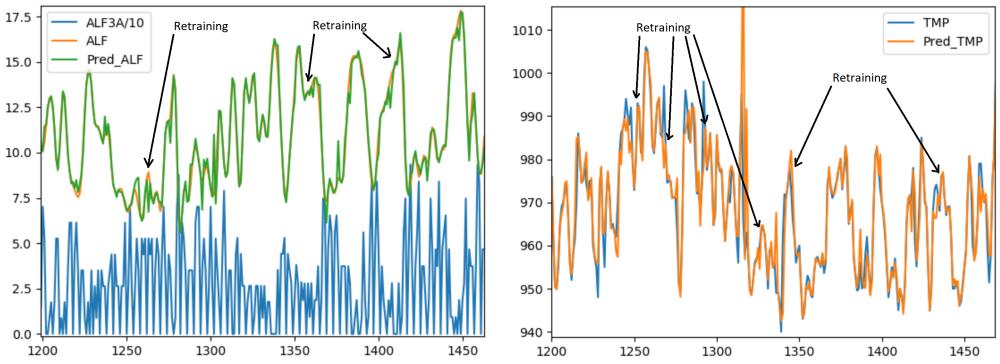


Figure 2: Simulation of bath chemistry control with retraining

4 Discussion and next steps

The linear model works well for multi-step ahead prediction, and jointly with neural networks provides good predictions. On the other hand, while the nonlinear dynamics still needs to be covered by a neural network, the training time remains a problem in implementing retraining in a cheaper hardware. For multi-step predictions, that imply a simulation of the plant in feedback, there is a significant loss in the accuracy the longer the prediction window. Provided that the cell dynamics is subject to change from time to time, one must consider also the possibility of not only retraining the model, but reprocess the clustering algorithm. In addition, eventual process disturbances must be included in the simulation to give an error estimation. The new generated data, including spatial variations did not seem to improve qualitatively the accuracy for multi-step ahead predictions. In conclusion, it is worth trying to review the cluster groups in the online operation in conjunction with the neural network training. Also, the approach of linear+neural can be applied for cell-based models as well, where a cluster model learns the general characteristics of that cluster while a cell model learns the cells particularities and operation.

References

[1] C. A. P. Braga and J. V. da Fonseca Netto. "A dynamic state observer to control the energy consumption in aluminium production cells". In: *Systems Science & Control Engineering* 4.1 (2016), pages 307–319. DOI: 10.1080/21642583.2016.1238325.

[2] M. Dupuis. "Mathematical Modelling of Aliumnium Reduction Cell Potshell Deformation." In: *Essential Readings in Light Metals*. Edited by G. Bearne, M. Dupuis, and G. Tarcy. Cham: Springer International Publishing, 2016, pages 439–444. ISBN: 978-3-319-48156-2. DOI: 10.1007/978-3-319-48156-2_63.

[3] L. Fortuna et al. *soft Sensors for Monitoring and Control of Industrial Processes*. London, UK: Aluminium Verlag, 2007. ISBN: 978-1-84628-480-9.

[4] K. Grjotheim and H. Kvande. *Introduction to Aluminium Electrolysis*. Düsseldorf, Germany: Aluminium Verlag, 1993. ISBN: 978-3-410-22027-5.

[5] M. Horvath and E. Vircikova. "Data Mining Model for Quality Control of Primary Aluminium Production Process". In: *Management and Production Engineering Review* 3.4 (2012), pages 47–53. DOI: 10.2478/v10270-012-0033-x.

[6] F. A. N. de Lima, A. M. F. de Souza, F. M. Soares, D. L. Cardoso, and R. C. L. de Oliveira. "Clustering Aluminum Smelting Potlines Using Fuzzy C-Means and K-Means Algorithms". In: *Light Metals 2017*. Edited by A. P. Ratvik. Cham: Springer International Publishing, 2017, pages 589–597. ISBN: 978-3-319-51541-0. DOI: 10.1007/978-3-319-51541-0_73.

[7] V. G. Pereira, R. C. D. Oliveira, and F. M. Soares. "Fuzzy Control Applied to Aluminum Smelting". In: *Fuzzy Logic*. Edited by E. P. Dadios. Rijeka: IntechOpen, 2012. Chapter 13. DOI: 10.5772/35452.

[8] T. Taskaya-Temizel and M. C. Casey. "A comparative study of autoregressive neural network hybrids". In: *Neural Networks* 18.1 (2005), pages 781–789. DOI: 10.1016/j.neuralnet.2005.06.003.

Real-time Power Monitoring for Heterogenous Data Centers

Lawrence Benson, Fabian Paul, Christian Werling, and Fabian Windheuser

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@student.hpi.de

Energy consumption is one of the major cost factors when operating a data center. Hence, monitoring the energy consumption of a data center is crucial for operation. This requires a comprehensive overview of the energy distribution among servers and running jobs. We propose a power monitoring system for heterogenous data centers. The system collects resource and power consumption metrics on data center-, node-, and process-level. It supports data center operators in monitoring the total energy consumption, detecting jobs with high energy profiles, and planning the allocation of new resources. We deploy and evaluate the system on the HPI Future Soc Lab, an on-site data center providing computational power to the research community.

1 Introduction

Power consumption in a data center is an emerging topic [6]. The increasing power demands require careful management and monitoring of data center resources. This brings new requirements for tools and services that support data center operators in maintaining a data center, such as energy-awareness and efficient resource utilization.

The HPI Future SOC Lab is an on-site data center equipped with latest high-performance servers donated by industry partners. It aims at providing its computational power to the research community around the world. In the semi-annual Call-for-Projects, researchers can apply for access to these computational resources.

Naturally, the Future SOC Lab hosts a highly heterogenous IT infrastructure. Furthermore, the Lab has a hard limit on its power consumption. When exceeded, machines fail in an unsafe and impractical manner. As the Lab's workload is generated by many individual researchers at unknown points in time, active power monitoring is necessary to prevent power exceedance and to identify especially demanding projects. Moreover, for economic and ecological reasons unnecessary machines should be turned off if not used for a longer duration.

To address these challenges, we implement a power monitoring system that collects resource utilization metrics from the whole data center and makes them easily accessible for data center operators. Based on these metrics, we provide tools to monitor the resource and power consumption on the data center-, node-, and project-level. The collected metrics can further be used to support planning for resource allocation and extending the data center.

2 Related Work

In this section, we present work most pertinent to the discussion of this paper in the field of power management. Similar to our approach, a lot of research was conducted to estimate power consumption in virtual machines. In this field, Zhao et al. [5] present a power monitoring solution for virtual machines to save energy during provisioning. We derive our power calculation from their linear equation describing the CPU consumption. Contrary to Contreras et al. [3], who use internal hardware counter as reference for throughput and load, we employ coarser activity metrics such as the CPU load. This allows us to utilize our system on different processor architectures without relying on specific hardware features. Another aspect in energy management, is treating it as scheduling resource. Specifically Chase et al. [2] further investigate this behavior and show a decrease in energy consumption while meeting a specified load. Finally, one of our important constraints, running on heterogeneous hardware, is discussed in [1] where Beloglazov et al. try to find optimal placements of virtual machines. Hereby, our approach is able to monitor the power consumption on a process level which also includes a different set applications besides virtual machines.

3 Contribution

In this section, we present our contribution towards real-time power monitoring in heterogenous data centers. First, we give a high-level system overview of the system (section 3.1), followed by more detailed explanations of the dashboards (section 3.2) and the exporters in section 3.3.

3.1 System Overview

In this section, we give a high level overview of our proposed system. Figure 1 shows the connections between all parts of the system, which is spilt into two main parts – one master node and multiple worker nodes. The master is run as a Docker container [8] on a dedicated server. The workers are nodes in the network that should be monitored. It is important to note here that our worker process should have very little CPU/memory impact on their host machine, as these are the production machines that should be utilized by the end user's jobs.

On the master node, we run Grafana[1] Prometheus[2] and a custom registry service. Grafana is used to display the power-usage-related dashboards to the system administrator. It queries the relevant data from the Prometheus instance running on the same server. We discuss the used dashboards and the interaction with Prometheus in more detail in section 3.2. We chose Prometheus for collecting data, as it is a widely

[1] https://grafana.com (last accessed 2020-04-01).
[2] https://prometheus.io (last accessed 2020-04-01).

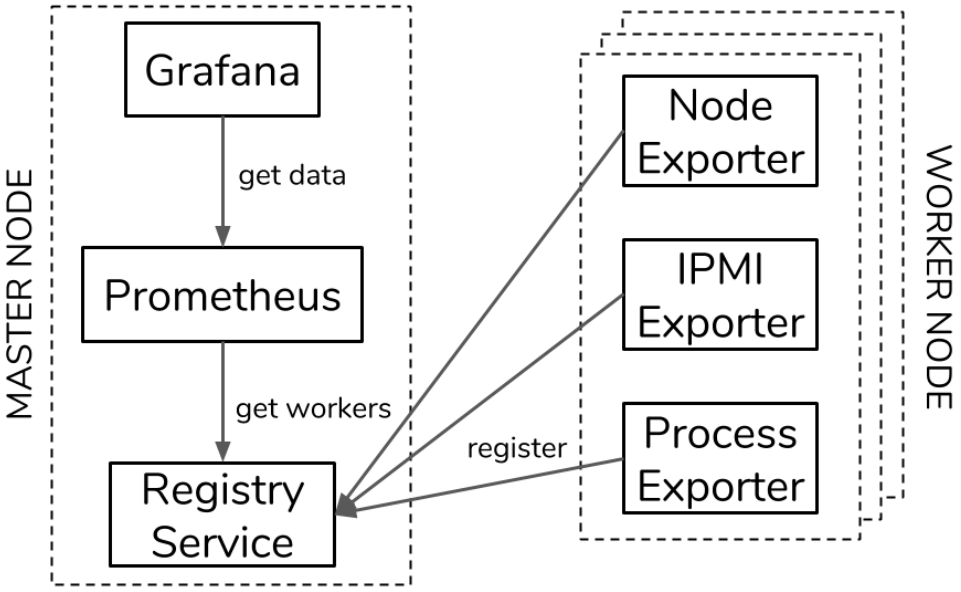


Figure 1: Architecture Overview of the Power Monitoring System

adopted tool for monitoring with a large community and stable releases. Prometheus is a pull-based time series database in which we store our metrics such as CPU load, power consumption, or CPU temperature. It allows for dynamic queries on the time series, so the user can view data from the last 5 minutes or the past month. On top of that, it provides an API for filtering and aggregating the data, which we heavily use. We describe the use of Prometheus together with the dashboards in section 3.2 and its interaction with the data exporters in section 3.3.

The workers consist of a few *exporters*, as they are called in Prometheus terminology, which are simple web servers that are queried for current data every few seconds. The data that is sent can be arbitrary, as long as it conforms to the Prometheus data format. For our use case, we chose three exporters, (1) the *node exporter*, provided by Prometheus itself, which exports some general CPU information, (2) an IPMI exporter that exports power information according to the IPMI protocol, and (3) a custom process exporter, which provides information about the current processes on each machine. We explain all of these exporters in more detail in section 3.3.

With the master and worker nodes in place, we can actively monitor the power consumption of the users on each machine in the data center. This allows the administrator to determine which users are causing a higher power consumption than expected. In the following section 3.2 and section 3.3, we give a more detailed explanation of the components in the system and how they work together.

3.2 Power Monitoring Dashboards

In this section, we present the Grafana dashboards used to monitor the power consumption of the the data center's servers. To get data from the Prometheus database

to the Grafana dashboards, we use Grafana's built-in Prometheus API. This allows us to formulate complex queries against the stored data. Supported operations include *joins*, *filtering*, and *aggregation*. The main advantage of this is that these potentially expensive queries are performed directly on the data and not sent to the front-end in advance.

There are two main views in our system, one that gives a general overview of all servers (section 3.2.1) and one that dives deeper into the state of a single server (section 3.2.2).

3.2.1 Multi-Server View

The multi-server view provides the administrator with some information about all servers.

Server IP	Avg	Min	Max	Current
192.168.42.168:9290	65 W	65 W	67 W	65 W
192.168.42.23:9290	92 W	88 W	96 W	89 W
192.168.42.56:9290	660 W	648 W	672 W	660 W

Figure 2: Power Consumption Statistics for Individual Servers

Figure 2 shows a central view of the administrator's dashboard. It gives a quick overview of the servers and their average, maximum, minimum, and current power usage for the selected time frame. We see that the first two servers are running with low power usage (<100 W) at the moment, whereas the third server is using more than seven times as much power (>600 W) compared to the second highest one.

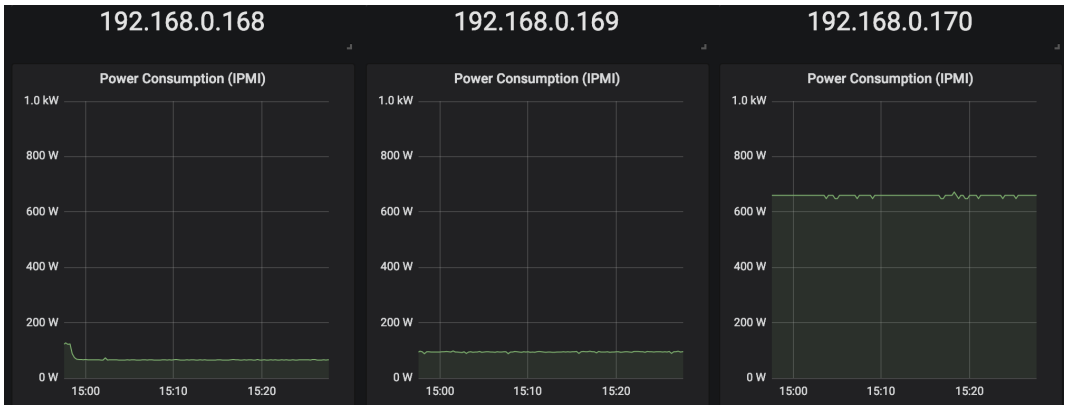


Figure 3: Dashboard with power consumption in Watts of three servers

Figure 3 shows the current power consumption of each server (as in figure 2) but as a graph, to detect trends more easily. It is possibly to scale the time frame

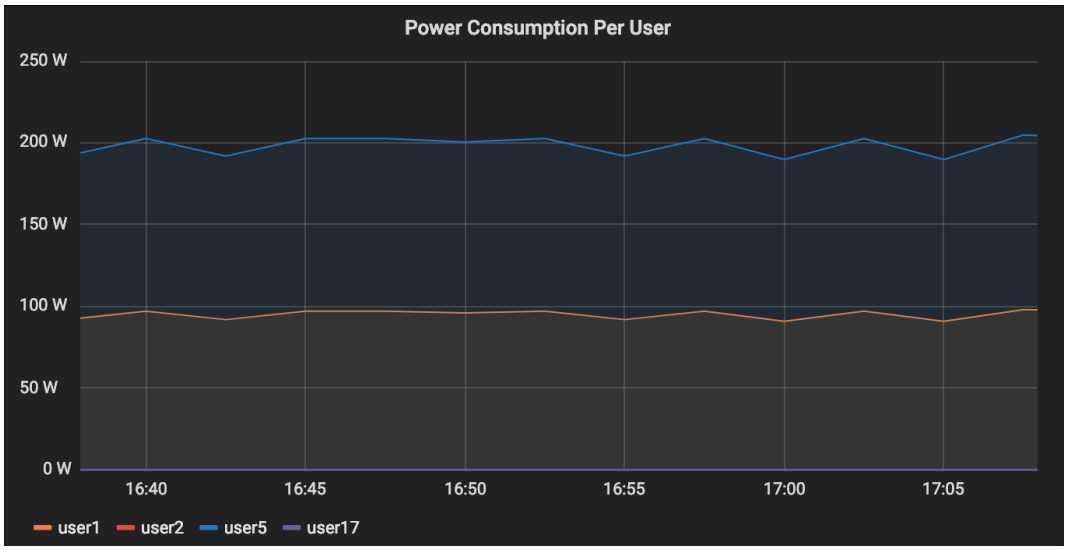


Figure 4: Dashboard with power consumption in Watts per user

arbitrarily, allowing the administrator to look at, e.g., the last five minutes or the last month. There are a few more dashboards in this multi-server view, which are omitted due to space concerns. They include CPU load, CPU clock frequency, CPU temperature.

3.2.2 Single-Server View

The single-server view gives the administrator more detailed information for a single server.

In figure 4, we see the power consumption per user on a single server. There are four active users on the host machine, *user1*, *user2*, *user5*, and *user17*. With roughly 200 W used, the administrator can see that *user5* is running a highly power-consuming application. An additional 100 W are consumed by *user1*, whereas *user2* and *user17* are idle on the server. This allows the administrator to make decisions on whether this consumption is acceptable for long-running jobs or not, achieving higher transparency in the overall power usage of the data center.

To calculate the power consumption per user, we take multiple metrics into account, such as system power consumption, the user's processes, and CPU load. To map these to an accurate wattage, we use a modified version of the function in [5].

$$E_{user} = \frac{E_{above\_idle}}{u_{cpu}(user)} \tag{1}$$

$$E_{above\_idle} = min(E_{sys}(T - 30)) \tag{2}$$

Here, E_{user} is the energy consumption per user that we want to display. To calculate this value, we look at the energy consumption above the system's idle (E_{above_idle}), which we define as the minimum recorded consumption of the previous 30 days ($E_{sys}(T - 30)$). With $u_{cpu}(user)$, we describe the total CPU utilization per user on a given server.

As in the multi-server view, there are more dashboards here that we omit for space reasons. They include, among others, CPU load per process, memory per user, and power consumption above system idle.

3.3 Power Monitoring Exporters

We use Prometheus to collect the necessary data in a pull-based manner, i.e., the Prometheus server requests the current data from all exporters every few seconds. An important distinction between push- and pull-based implementations needs to be made here. There has been a lot of discussion regarding this topic [4, 7] and both options have advantages and disadvantaged, ranging from better manageability to resource efficiency. We argue, however, that for a small data center, as we are running, the advantage of central management via one application that pulls the data outweighs the potential reduced resource consumption, as proposed in [7].

In order for Prometheus to pull the data, it needs to be aware of all the nodes in the data center. For this, we implemented a light-weight registration service[3] that provides a single HTTP endpoint, which allows new workers to register themselves. The registration service then provides Prometheus with the list of current workers that need to be queried.

Each worker consists of three exporters. An exporter is a simple HTTP endpoint that delivers current values in a specific format that Prometheus can read, upon request. This allows the administrator to determine when and how often the values for each exporter need to be read. The *node-exporter*[4] is provided by Prometheus itself. It exports multiple OS-level and hardware metrics, including CPU load, clock frequency, and temperature. We need this information to get a general overview of each host. The *IPMI-exporter*[5] is provided by SoundCloud.[6] It exports power information for hosts with an Intelligent Platform Management Interface (IPMI). This is the core information for our dashboards, due to the strong focus on power consumption. Our custom *ps-exporter*[7] serves information regarding all processes and their respective users. This information is required for a detailed breakdown of power consumption per-user and per-process. All three exporters are wrapped in separate Debian Packages, so they can also be used independently of your system.

4 Conclusion and Future Work

We implement and deploy a power monitoring system for heterogeneous data centers. By collecting multiple resource utilization metrics from across the data center,

[3] https://github.com/osmhpi/power-tools/tree/master/exporter-registry (last accessed 2020-04-01).
[4] https://github.com/prometheus/node_exporter (last accessed 2020-04-01).
[5] https://github.com/soundcloud/ipmi_exporter (last accessed 2020-04-01).
[6] https://soundcloud.com (last accessed 2020-04-01).
[7] https://github.com/osmhpi/ps-exporter (last accessed 2020-04-01).

the system provides fine grained insights into the resource and power utilization of a data center. In the context of the Future SOC Lab, this system is used to monitor registered projects. It gives the administrator an overview over active and inactive projects and makes it possible to detect projects that abuse their project to waste resources. Furthermore, the system helps in the planning process of accepting projects and assigning resources by providing historical data from previous projects. This guides the administrator in predicting resource and power utilization of new resources and allows them to evenly distribute resource utilization and to plan the power budget.

Additionally, the centralized collection of multiple resource utilization metrics from the whole data center makes it possible to easily implement additional use cases. Based on the collected data, the administrator could set up alerts when the power consumption gets dangerously high and approaches the global power limit. This provides them with an early warning system and reduces maintenance overhead.

The collected power utilization metrics can also be used to extend the job scheduler for considering power consumption. The historical data can be used to estimate the power consumption of submitted jobs which then can be used to avoid scheduling jobs on machines which are close to their power limit.

References

[1] A. Beloglazov and R. Buyya. "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers". In: *Concurrency and Computation: Practice and Experience* 24.13 (2012), pages 1397–1420.

[2] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. "Managing energy and server resources in hosting centers". In: *ACM SIGOPS operating systems review* 35.5 (2001), pages 103–116.

[3] C. Gilberto and M. Margaret. "Power prediction for intel xscale processors using performance monitoring unit events power prediction for intel xscale processors using performance monitoring unit events". In: *ISLPED*. Volume 5. 2005, pages 8–10.

[4] H. Huang and L. Wang. "P&P: A combined push-pull model for resource monitoring in cloud computing environment". In: *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE. 2010, pages 260–267.

[5] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. "Virtual machine power metering and provisioning". In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, pages 39–50.

[6] J. G. Koomey. "Worldwide electricity used in data centers". In: *Environmental research letters* 3.3 (2008), page 034008.

[7] J.-P. Martin-Flatin. "Push vs. pull in web-based network management". In: *Integrated Network Management VI. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management.(Cat. No. 99EX302)*. IEEE. 1999, pages 3–18.

[8] D. Merkel. "Docker: Lightweight Linux Containers for Consistent Development and Deployment". In: *Linux J.* 2014.239 (Mar. 2014). ISSN: 1075-3583. URL: http://dl.acm.org/citation.cfm?id=2600239.2600241.

Big mobility data analysis, machine learning and sensor fusion for activity recognition

Dragan Stojanovic, Igor Djordjevic, Natalija Stojanovic, and Aleksandra Stojnev Ilic

Computer Science and Engineering Department
University fo Nis, Faculty of Electronic Engineering
{firstname.lastname}@elfak.ni.ac.rs

With the widespread use of mobile devices, such as smart phones, smart watches, bracelets, with more and more integrated sensors, massive amounts of data relevant to people health and activities are collected and stored, potentially in a cloud. In this report we present CitySensing platform and Apache Spark application for analysis and recognition of people activities based of sensor fusion and Spark MLlib machine learning library. The application provides processing and analytics of Big sensor data originated from smart personal devices with the aim to detect and recognize human activities and behaviour.

1 Introduction

The CitySensing project is devoted to research and development of methods, techniques, software systems and platforms for Big Data applications that provide processing, analysis and mining of large-scale data collected by mobile devices and Internet of Things in Smart Cities.

Plenty of Sensors, integrated in mobile and wearable devices, such as smart phones, watches, bracelets, etc., represent sources of large amount of data about user movements, activities and health conditions. Human daily activities and conditions can be recognised by analysis and mining of data originated from accelerometers, gyroscopes, audio sensors, location sensors, etc. Such information can be a key in monitoring and preserving people health and secure behaviour.Big data analysis and mining represents a big challenge and Apache Spark is one of the leading technologies to work with Big and distributed data.

The main focus of this work is to apply machine learning techniques for human activity recognition from mobile devices data and detect and evaluate appropriate classification algorithms. Such algorithm is the central component of applications that collect data from various mobile sensors and fuse them to obtain the most accurate results. The basic machine learning algorithms used are binary and multiclass logistic regression. This work use publicly available dataset of 800 MB in size, comprising data from smartphone and smartwatch senors for 30 volunteers over 60 days with extracted attributes and labels.

2 Background and related work

The analysis and health problem detection can be performed on a mobile device (phone, watch, bracelet, etc.) leveraging edge computing devices, or at private/public cloud computing infrastructure [2].

The fusion, analytics and mining of mobile sensor data, collected from personal mobile and health devices, as well as Smart city infrastructure can provide a personalized health and monitoring system for general well-being where individuals can be provided with healthcare tailored to their needs. Moreover, such system supports efficient collection and analysis of massive quantities of heterogeneous and continuous health and activities data (Big Data) from a group, or a crowd of users. The storage, aggregation, processing and analysis of Big health data could be performed within public, private or hybrid cloud infrastructure. The results of data analysis and mining are provided to physicians, healthcare professionals, medical organisations, pharmaceutical companies, etc. through tailored visual analytics, dashboard applications. There is a number of research and development challenges that must be addressed prior to wider application of such personalized healthcare systems that continuously monitor people's health and activities and respond appropriately on critical events and conditions. These include, but are not limited to security, privacy and data ownership, sensor data fusion, scalable algorithms and systems for analytics and data mining, and edge-cloud models for processing and analytics.

Khan et al. in [4] presented the latest research and development efforts and achievements in the field of smart healthcare regarding high performance computing (HPC) and large-scale healthcare architectures, data quality and large-scale machine learning models for smart healthcare, Internet-of-Things, fog computing, and m-Health, as well as wearable computing, Quality of Service (QoS), and context-awareness for smart healthcare.

With the development of Internet of Things (IoT) concepts, the idea of IoT healthcare systems has been an interesting topic for large number of researchers. Baker et al. in [1] have presented state-of-the-art research related to wearable IoT healthcare system. A standard model for application in future IoT healthcare system was proposed, along with an overview of existing challenges including security, privacy, wearability and low-power operation.

Riazul Islam et al. in [3] have given an overview of existing IoT–based healthcare network studies, and state-of-the-art network architectures, platforms, applications, and industrial trends in this area. Furthermore, they highlighted security and privacy issues and proposed a security model aiming to minimize security risk related to health care. Specifically, they presented an extensive overview of IoT healthcare systems, one of them being the Internet of m-Health Things (m-IoT): an operating symbiosis of mobile computing, medical sensors, and communications technologies for healthcare services.

An important part of IoT healthcare is obtaining insights from large data generated by IoT devices. The focus of [5] was on IoT architecture, opportunities and challenges, but from the data analysis point of view. In this paper a brief overview of research efforts directed toward IoT data analytics, as well as the relationship between big data

analytics and IoT were given. Furthermore, analytic types, methods and technologies for big IoT data mining are discussed.

In [6], authors have used smartphone and smartwatch sensors to recognize detailed situations of people in their natural behaviour. Initial tests were conducted over labelled data from over 300k minutes from 60 subjects. A dedicated application for data retrieval was implemented and presented. As stated in the paper, the main contribution is the emphasis on in-the-wild conditions, namely naturally used devices, unconstrained device placement, natural environment and natural behavioural content. The main challenge when recognizing context in non-controlled conditions is high diversity and variance of the data. However, it was shown that common mobile devices, in their natural usage, can capture information about a wide range of behavioural attributes.

3 Sensor fusion and Big Data mining for activity recognition

The CitySensing platform consists of mobile application components, server components and visualization/analytics components organized in a distributed architecture. It fully leverages processing, sensing and communication capabilities of mobile and wearable devices and provides distributed and scalable storage, processing, analysis and mining of sensor and mobility data at private/public cloud infrastructure.

High-level mobility and sensor data generated at users' mobile and wearable devices are fused, aggregated, processed and analyzed at the RemoteHealth server running on a cluster/cloud infrastructure (HPI Future SOC Lab cluster). For processing and analysis of Big mobility and IoT data, RemoteHealth server is based on distributed processing frameworks, such as MapReduce/Hadoop and Apache Spark.[1] For efficient processing and analysis of masssive mobility and IoT data are stored in a distributed file system in the cloud/cluster (HDFS). For analysis and mining of large-scale data, Spark MLlib library[2] is used. For communication between different RemoteHealth components Apache Kafka,[3] a distributed message platform is used.

We implemented Human Activity Recognition (HAR)application within RemoteHealth server, based on Apache Spark platform and Spark MLlib machine learning library. Our work is related to remote health and activities monitoring and development of large-scale data intensive cloud application for remote monitoring of people health and activities. Taking into account all the challenges related to health/activities monitoring data and control flows that should be employed in a public, private or hybrid cloud, we have developed Remote Health/Activities Monitoring architecture and implemented an application for analysis and mining of mobile/wearable sensor data.

[1] https://spark.apache.org/ (last accessed 2020-04-01).
[2] https://spark.apache.org/mllib/ (last accessed 2020-04-01).
[3] https://kafka.apache.org (last accessed 2020-04-01).

For the implementation of prototype system we used pre-recorded data from ExtraSensory dataset[4] generated by mobile health application for data collection provided by Extrasensory project. ExtraSensory dataset contains data from 60 users and more than 300K labeled examples (minutes) originated from smartphone and smartwatch sensors. The dataset is publicly available for context recognition research [7].

The dataset was obtained from sensor data retrieved from a smartphone and a smart watch: Accelerometer, Gyroscope, Magnetometer, Watch accelerometer, Watch compass, Location, Location (quick), Audio, Audio magnitude, Phone state (App status, battery state, WiFi availability, on the phone, time-of-day), and additional sensors if available (light, air pressure, humidity, temperature, proximity). Among them, we have used data from six sensors: smartphone accelerometer,gyroscope, watch accelerometer, location, audio, and phone state.

We have implemented single-sensor classifier based on features extracted from the raw sensor data which helps us to understand how informative each sensor can be, independent of the other sensors, for a given context label. We used logistic regression, a linear classifier that outputs a continuous value (interpreted as probability) in addition to the binary decision. This classifier is a base for sensor fusion classifiers also leveraging logistic regression (Figure 1), and these are:

- Early fusion (EF),
- Late fusion using average probability (LFA),
- Late fusion using learned weights (LFL).

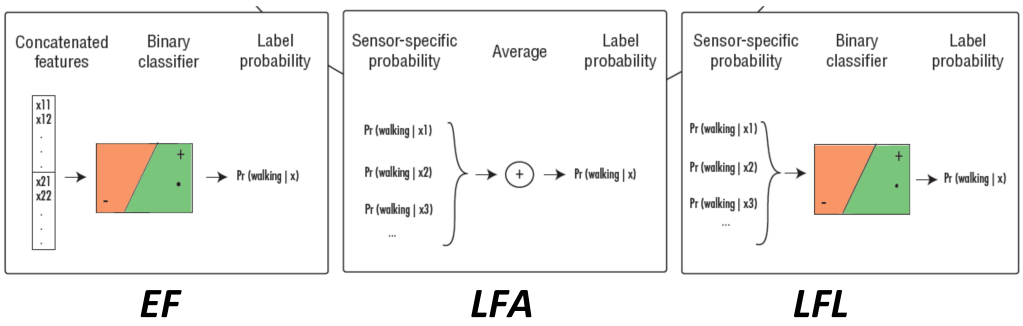


Figure 1: Sensor fusion classifiers based on logistic regression [6]

[4]https://extrasensory.ucsd.edu/ (last accessed 2020-04-01).

4 HAR application implementation

For the purpose of deploying, testing and evaluation of RemoteHealth Big data applications we use following Future SOC Lab infrastructure:

- Cluster of 9 virtual machines: 1 master (XL VM), 8 slaves (L VM) installed and configured with Apache Hadoop, Spark and Kafka.
- 1000 Core cluster, using MPI on Slurm.
- GPU using NVidia/CUDA over Docker.

The Human Activity Recognition (HAR) application was developed in Java and Spring framework and has been deployed on a cluster of 9 virtual machines, 1 master and 8 workers (Figure 2). The timeline of spark jobs and executors executed on cluster nodes is shown in Figure 3.

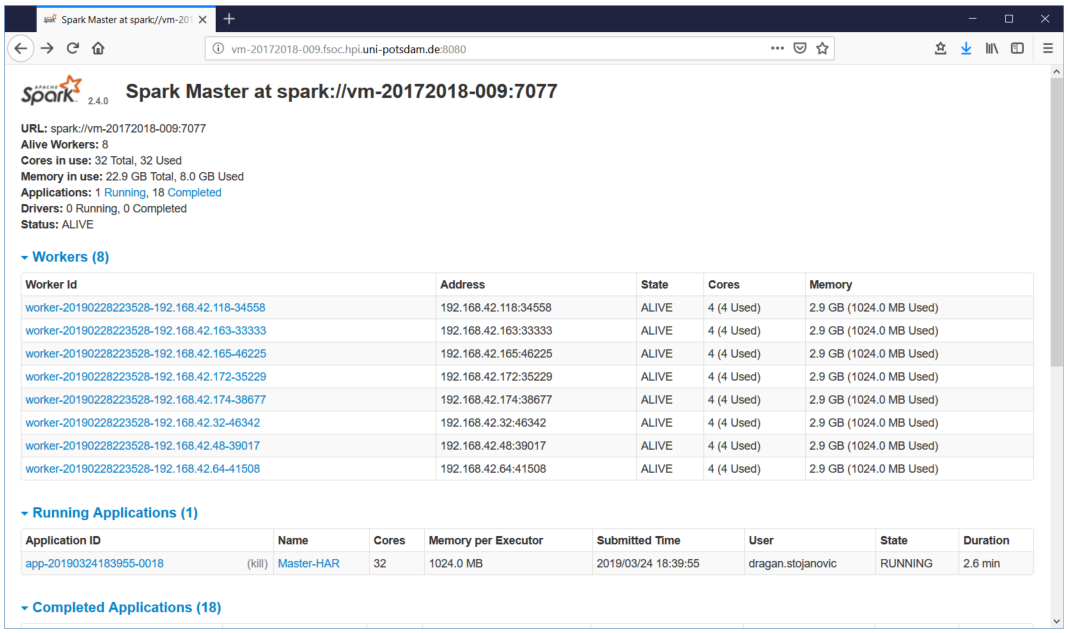


Figure 2: Spark application running over HPI FutureSOC Lab infrastructure

5 Evaluation

For the evaluation of sensor fusion classifiers for activity recognition, we have calculated standard accuracy measures: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Based on the we calculated the following performance metrics> Accuracy, Sensitivity (TPR), Specificity (TNR), Precision, Balanced Accuracy (BA) and F1 measure. The results shown in Figure 4 demonstrate

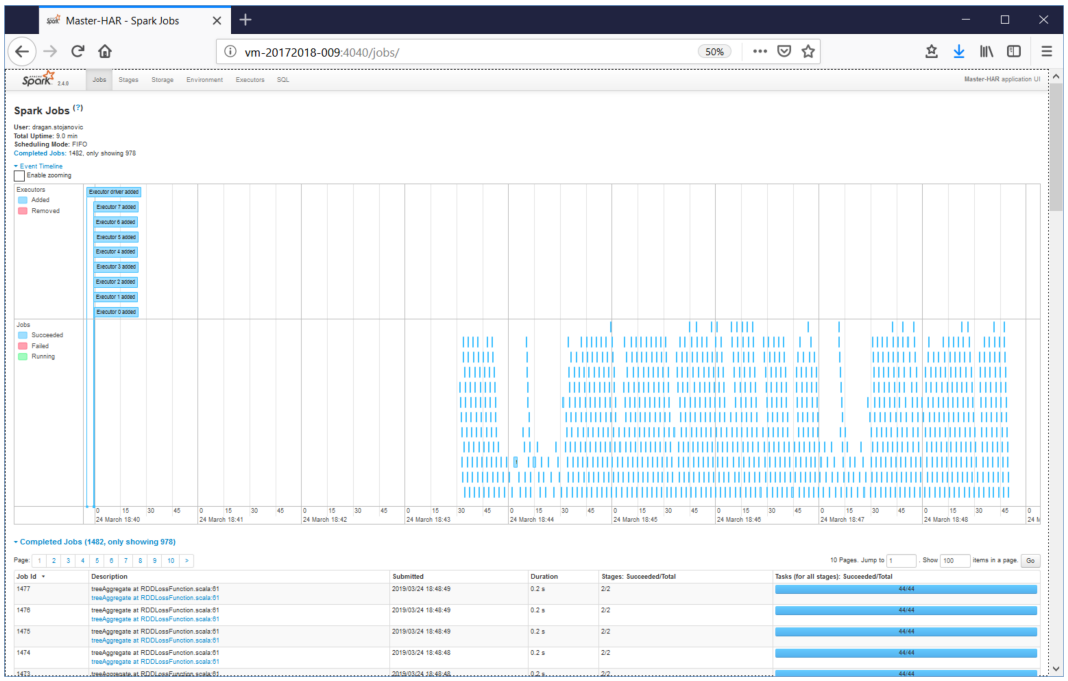


Figure 3: The timeline of Spark jobs execution

the benefits of applying sensor fusion and particularly late fusion classifier methods, LFA and LFL.

The accuracy of activity recognition for various combination of sensors is evaluated on the part ExtranSensory data set annotated with labels: ”FIX_walking”, ”FIX_running”, ”BICYCLING”, ”SITTING”, ”LYING_DOWN”. The confusion matrices for different combination of sensors are presented in Figure 5, showing the best classification results when using all sensors in classification algorithm.

6 Conclusion

We expect that availability of Future SOC Lab infrastructure for our project will further improve our research expertise and experience in domains of mobile sensing, IoT and Big mobility, activity and health-related data processing, analysis and mining and will result in scientific achievements through preparation/publication of technical re-ports, scientific papers and development of prototype/demo solutions.

The availability of HPI Future SOC Lab infrastructure gives us useful insights in deploying and execution of Big data applications on the real cloud infrastructure.

	Accuracy	Sensitivity	Specificity	BA	Precision	F1 score
Accelerometer	0,697	0,573	0,694	0,634	0,173	0,222
Gyroscope	0,657	0,584	0,662	0,623	0,156	0,198
Watch Accelerometer	0,701	0,654	0,699	0,674	0,176	0,234
Location	0,683	0,566	0,678	0,621	0,189	0,231
Audio	0,676	0,654	0,672	0,663	0,176	0,255
Phone state	0,709	0,678	0,703	0,691	0,190	0,246
Early fusion (EF)	0,875	0,571	0,882	0,726	0,262	0,343
Late fusion – LFA	0,775	0,746	0,771	0,757	0,225	0,297
Late fusion – LFL	0,817	0,697	0,819	0,758	0,244	0,313

Figure 4: Accuracy of activity recognition

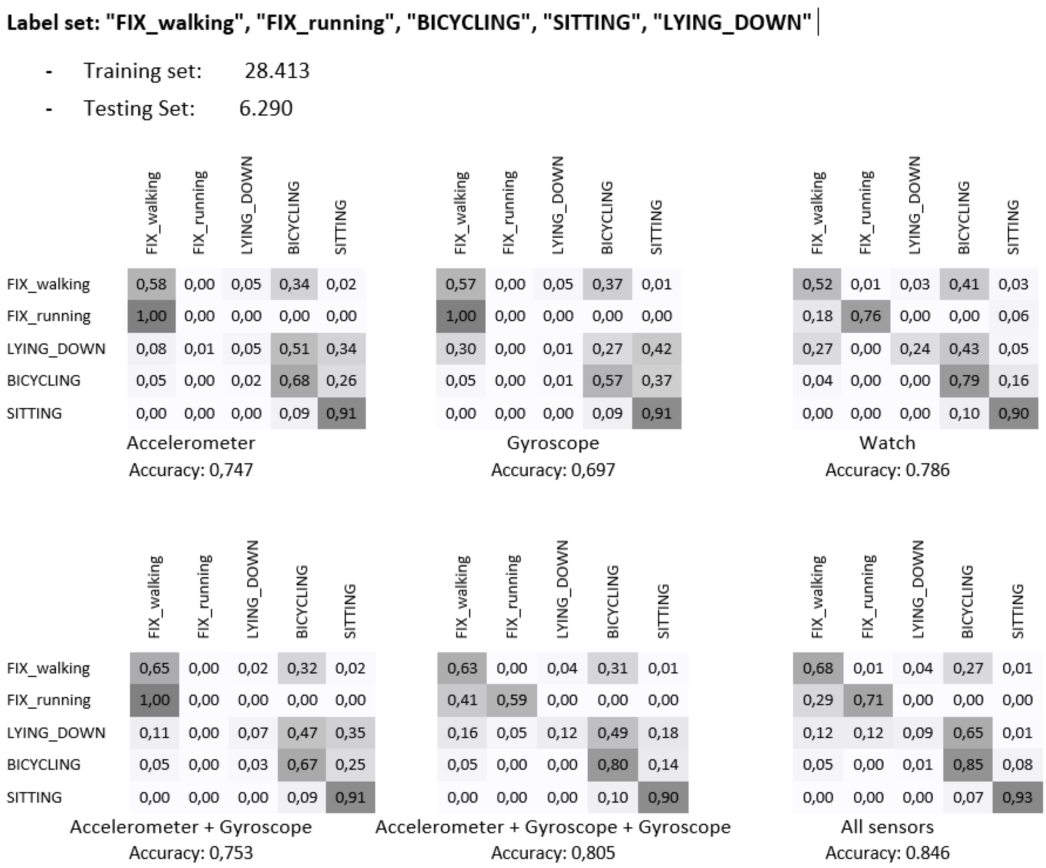


Figure 5: Confussion matrices for classifiers

References

[1] S. B. Baker, W. Xiang, and I. Atkinson. "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities". In: *IEEE Access* 5 (2017), pages 26521–26544. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2775180.

[2] H. Dubey, A. Monteiro, N. Constant, M. Abtahi, D. Borthakur, L. Mahler, Y. Sun, Q. Yang, U. Akbar, and K. Mankodiya. "Fog Computing in Medical Internet-of-Things: Architecture, Implementation, and Applications". In: *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Springer International Publishing, 2017, pages 381–321. ISBN: 978-3-319-58280-1.

[3] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. Kwak. "The Internet of Things for Health Care: A Comprehensive Survey". In: *IEEE Access* 3 (2015), pages 678–708. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2437951.

[4] S. U. Khan, A. Y. Zomaya, and A. Abbas, editors. *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-58280-1. DOI: 10.1007/978-3-319-58280-1.

[5] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob. "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges". In: *IEEE Access* 5 (2017), pages 5247–5261. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2689040.

[6] Y. Vaizman, K. Ellis, and G. Lanckriet. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches". In: *IEEE Pervasive Computing* 16.4 (Oct. 2017), pages 62–74. ISSN: 1536-1268. DOI: 10.1109/MPRV.2017.3971131.

[7] Y. Vaizman, K. Ellis, G. R. G. Lanckriet, and N. Weibel. "ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Edited by R. L. Mandryk, M. Hancock, M. Perry, and A. L. Cox. ACM, 2018, page 554. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174128.

Behavior-based authentication

Feature engineering and performance evaluation based on large user profiles

Vera Weidmann[1] and Leon Lowitzki[2]

neXenio GmbH
vera.weidmann@nexenio.com, leon.lowitzki@nexenio.com

1 Introduction

Our project contributes to the growing interest of mobile and behavior-based authentication systems. Next to fingerprints or facial features, another meaningful authentication method is the person's gait pattern. Since our smartphones are equipped with a variety of sensors, they can measure the environment and behavior surrounding the phone. We carry these phones with us around the clock, even closely attached to our bodies. Accordingly, data corresponding to the body movement is generated. Research shows that these recorded data signals can be shaped to a unique signature for the phone's holder. The authentication accuracy, though, is highly dependent on users' physiological and environmental circumstances. The more situations are covered by the user's training state, the more the system's precision is challenged. In this context, statistical investigations and machine learning algorithms are applied.

Our research team at neXenio confronts behavior, in particular gait-based authentication, by a enormous data set, covering six thousand walking sequences that have been recorded over the last years.

2 Behavior-based authentication

Nowadays, smartphones can be unlocked via a password, the own fingerprint, or, as more recently introduced, via face recognition. The last two methods are biometric authentication methods, which use the user related characteristics, e.g. the friction ridges of the finger or facial features, to recognize people.

Nowadays, another biometric generates interest and enthusiasm: gait authentication. Here, the user's motions, such as the leg's forward, backward or sideways movements, are observed by sensors. The next sections introduce the general methodology for gait-authentication as well as neXenio's course of action in this field.

2.1 Methodology for gait-based authentication

Gait-based authentication methods are well researched nowadays. Mostly, walking sequences, recorded by smartphones or smart watches' inbuilt sensors, such as accelerometer and gyroscope sensor, are reflected. Sprager and Juric [9] as well as Connor [2] summarized the state of the art methodologies used in this field. Guidelines for general data processing are stated by [1] and [3].

In brief, a walking sequence is cut in pieces and summarized by an average gait cycle. This cycle is used as a template and the dissimilarity of a new step is concluded by different distance metrics, such as Euclidean, Hamming, Manhattan or Tanimoto distance. Other approaches use statistical measurements in the time and frequency domain to gather descriptive information about walking sequences. Calculations as mean, median, standard deviation, the third and fourth order cumulants, the skewness and kurtosis, distribution parameters, such as a ten-bin histogram are used. [5, 7, 8] Likewise, we found features as the root mean squared, median absolute deviation, average absolute variation, quantiles, the interquartile range, correlations and zero-crossing in several papers. In the frequency domain, Fourier and cepstral coefficients, discrete cousin transformations and wavelets are used [9]. Next to principal component analysis, Gait Dynamic Images [12] or geometric template matching were applied to gait sequences [4].

Besides cycle-based assignments, different machine learning techniques are used for authentication: linear and logistic regressions, k-nearest neighbors, support vector machines (SVM), hidden markov models and convolutional neuronal networks.

2.2 BAuth @ neXenio

neXenio GmbH is a small company located in the middle of Berlin. They take up the challenge to develop high secure IT-solutions that address data sharing and virtual collaboration needs of digital workspaces. One of their products is directed to the idea of **B**ehaviour-based **Auth**entication by their product called "BAuth". The most important difference to other authentication systems and research approaches is that neXenio's implementation comprises the premise of data privacy and security. Considering that data will never be sent to external computation resources, data is processed locally on the device itself. Thus, the underlying classification approach is required to be an one-classification problem. An algorithm is deployed that only considers data from a single person. This training set is neither polluted by any outlier nor is enriched by data from other users. The algorithm must detect whether a new unknown walking observation fits to the learned pattern. In general, this type of classification shows less precise results than binary or multi-class classification as the difference between users are not known. Only very few research studies considered this classification type, e.g. by using one-class SVMs.

Another challenge regarding local processing is the device's battery drain. Therefore, neXenio implemented a more efficient outlier technique instead of widely utlilized battery-expensive algorithms. This outlier recognition algorithm is based on multi-level hierarchical nested histograms. Each histogram creates a discrete

frequency distribution of the sensor data's processed features to a specific grain. Features, mentioned in the previous section 2.1, were evaluated and assembled in a way that best authentication performances were gained.

3 The Problem of learning from a variety of gait patterns

Real-world gait-based authentication applications have to deal with the variety of natural gait forms. If a specific, but genuine form is unknown by the algorithm, it can hardly be assigned to the user. Gait-affecting factors are of physiological or environmental nature. While physiological factors induce more unconscious circumstances, such as permanent gait abnormalities or temporal changes caused by mood, environmental factors are represented by clothing, shoes, surfaces, slopes or obstacles [9]. Real world behavior-based authentication systems need to be robust for long-term utilization as these factors can change to varying degrees from time to time.

Public datasets for gait recognition do not imply a comprehensive overview about user's possible walking situations. While OU-ISIR Biometric Database of Osaca University [6] is the largest database in the field of gait recognition, holding nearly 750 subjects, just one environmental factor, inclined terrain, was recorded in one session. Frank et al. [4] published another dataset in cooperation with the McGill University. This database contains just 20 participants, but data is recorded in the wild in two distinct sessions, meaning that people could have worn different clothes and shoes per session. Research that relies on this database, detected the effect of clothes the most. In all analyses the authentication performance suffers in cross-day comparison, particularly where people had a major change in trousers' type. [4, 10]. This effect is also shown by the larger dataset of Subramanian et al. [11].

Purpose of Future SOC Lab utilization

In the last years, we collected a great amount of data files, covering this variety of physiological and environmental forms. At the moment, our statistical investigations as well as our novelty detector, which we use for user classification, can be just tested by dividing our dataset into smaller subsets. An evaluation that comprises all walking circumstances is not feasible as the limits of our machines according to RAM are reached.

By applying for utilization of the HPI Future SOC Lab we aimed to improve our authentication approach, refine feature engineering and modeling in regard to meet stability for wide user profiles.

- Calculation of performance metrics of the current state algorithm by using the total data amount of six thousand walking samples.
- Evaluation of best feature combinations and model parameters. Strategies for model selection, e.g. by best subset or forward stepwise selection, can be accelerated by parallel computing.

4 Future SOC Lab resources

Future Soc Lab provided us with an isolated server, allowing us to process all of our data in parallel. In total, we run 8 tasks, including the initial evaluation of our current authentication implementation (Job 1) as well as feature evaluation (Job 2–4) and parameter tuning (Job 5–7).

To reduce computation time, time-expensive feature selection was split into two processes. First of all, the performance of single features was evaluated. Subsequently, just best features were combined and different model parameters were tuned. Table 1 summarizes RAM and time consumption for each task.

Table 1: Future SOC Lab Resources

Job#	Description	RAM computation	Time
1	Evaluation of current implementation	16 GB	1 h 40 min
2	Feature calculation	17 GB	6 h
3	Single feature selection	17 GB	7 h 30 min
4	Feature combination selection	19 GB	23 h 30 min
5	Parameter tuning 1	19 GB	71 h
6	Parameter tuning 2	19 GB	33 h
7	Parameter tuning 3	19 GB	48 h 50 min
8	Evaluation of new implementation	16 GB	1 h 40 min

5 Evaluation

5.1 Performance metrics

For classification objectives, generally, the true positives, false positives, false negatives and true negatives are calculated for evaluating the performance of an algorithm. Based on these four performance classes, certain metrics can be calculated: accuracy, recall, precision, specificity and the F1-score. The latter metric is a combination of recall, how often a genuine walking sequences has been classified correctly over all real positives, and precision, the ratio of correct predicted positive observations to the all predictions.

Additionally, the equal error rate (EER) is included in model selection. This metric reflects the closest point of false matches and false non-matches. While false matches are equivalent to false negatives, where imposters got authenticated, false non-matches include all false positives plus sequences where nobody could get authenticated.

In addition, neXenio uses another measurement, the fitness score, which is based on a fitness function to measure goal achievement. If all genuine walking sequences are classified correctly and imposters are discarded the fitness score will return 1.

5.2 Findings

The current implementation of our authentication model shows an EER of 6 %. Compared to other research studies this error rate is already satisfying. However, reflecting general classification metrics, such as the F1-score (49 %), it shows that the model needs improvements to ensure robust authentication. Here, a recall of 100 % implies that all genuine users could be authenticated. Nevertheless, a precision of 42 % declines the F1-score, meaning that too many imposters got authenticated. The fitness score of our present implementation is 67 %.

With feature evaluation and model selection we improved the F1-score by 5 % and the fitness score by 4 %. We gained an EER of 4 %. Additionally, this performance is reached by including less features in the model than before. This means, we also could reduce device's battery consumption, since less features have to be calculated.

5.3 New challenges

Although we could improve all performance metrics, we realized that some users had less false positives than others. This might be caused by the training size or still by gait variations included in the user's specific training set. Training sets needs to be analyzed in detail. Consequentially, a next challenge is to determine how we can adjust training states so that they can evolve with the user's walking condition.

6 Conclusion and future work

By using resources of HPI Future SOC Lab we analyzed and improved our authentication model. For the first time we conducted an evaluation that comprises all walking sequences that were recorded over the last years. New features and model parameters were analyzed and best combinations could directly be implemented to our application. All metrics, as F1-score, EER and fitness score could be improved.

In the future, we aim to refine our authentication approach by more feature engineering. We want to acquire new users, enrich the user's profile by new walking samples and additional motions and combine our data collection with publicly accessible data files. In this regard, we are looking forward to using the resources of Future SOC Lab in the next semester again. We want to thank the Future SOC Lab team, and in particular Bernhard Rabe, for the great support.

References

[1] A. Buriro, Z. Akhtar, B. Crispo, and S. Gupta. "Mobile Biometrics: Towards A Comprehensive Evaluation Methodology". In: *2017 International Carnahan Conference on Security Technology (ICCST)*. August. 2017. ISBN: 978-1-5386-1585-0. DOI: 10.1109/CCST.2017.8167859.

[2] P. Connor and A. Ross. "Biometric recognition by gait: A survey of modalities and features". In: *Computer Vision and Image Understanding* 167 (2018), pages 1–27. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2018.01.007.

[3] R. Ferrero, F. Gandino, B. Montrucchio, M. Rebaudengo, A. Velasco, and I. Benkhelifa. "On gait recognition with smartphone accelerometer". In: *Proceedings - 2015 4th Mediterranean Conference on Embedded Computing, MECO 2015 - Including ECyPS 2015, BioEMIS 2015, BioICT 2015, MECO-Student Challenge 2015* (2015), pages 368–373. DOI: 10.1109/MECO.2015.7181946.

[4] J. Frank, S. Mannor, J. Pineau, and D. Precup. "Time Series Analysis Using Geometric Template Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3 (Mar. 2013), pages 740–754. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.121.

[5] T. Hoang, D. Choi, and T. Nguyen. "Gait authentication on mobile phone using biometric cryptosystem and fuzzy commitment scheme". In: *International Journal of Information Security* 14.6 (2015), pages 549–560. ISSN: 1615-5270. DOI: 10.1007/s10207-015-0273-1.

[6] T. T. Ngo, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi. "The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication". In: *Pattern Recognition* 47.1 (2014), pages 228–237. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2013.06.028.

[7] C. Nickel. "Accelerometer-based Biometric Gait Recognition for Authentication on Smartphones". PhD thesis. Darmstadt: Technischen Universität Darmstadt, June 2012. URL: http://tuprints.ulb.tu-darmstadt.de/3014/.

[8] C. Nickel and C. Busch. "Classifying accelerometer data via hidden Markov models to authenticate people by the way they walk". In: *IEEE Aerospace and Electronic Systems Magazine* 28.10 (2013), pages 29–35. ISSN: 0885-8985. DOI: 10.1109/MAES.2013.6642829.

[9] S. Sprager and M. Juric. "Inertial Sensor-Based Gait Recognition: A Review". In: *Sensors* 15.12 (Sept. 2015), pages 22089–22127. ISSN: 1424-8220. DOI: 10.3390/s150922089.

[10] S. Sprager and M. B. Juric. "An Efficient HOS-Based Gait Authentication of Accelerometer Data". In: *IEEE Transactions on Information Forensics and Security* 10.7 (2015), pages 1486–1498. ISSN: 1556-6013. DOI: 10.1109/TIFS.2015.2415753.

[11] R. Subramanian, S. Sarkar, M. Labrador, K. Contino, C. Eggert, O. Javed, J. Zhu, and H. Cheng. "Orientation invariant gait matching algorithm based on the Kabsch alignment". In: *2015 IEEE International Conference on Identity, Security and Behavior Analysis, ISBA 2015* (2015), pages 1–8. DOI: 10.1109/ISBA.2015.7126347.

[12] Y. Zhong, Y. Deng, and G. Meltzner. "Pace independent mobile gait biometrics". In: *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems, BTAS 2015* (2015), pages 1–8. DOI: 10.1109/BTAS.2015.7358784.

Overfitting on purpose to design new algorithms

Markus Wagner

Optimisation and Logistics Group
University of Adelaide, Australia
{firstname.lastname}@adelaide.edu.au

Back in 2017, the teams around Prof. Tobias Friedrich (HPI, Chair for Algorithm Engineering) and Dr. Markus Wagner (University of Adelaide, Australia) have explored the concept of automated algorithm configuration to design new search operators. The experiments inspired theoretical investigations, which proved that the new methods beat state-of-the-art. In this project, we adopt this idea to investigate the so-called Travelling Thief Problem – a problem with interdependent components.

1 Project Idea

1.1 Data-Driven Search-Based Software Engineering

Data-Driven Search-Based Software Engineering (DSE) [3] combines insights from Mining Software Repositories (MSR) and Search-based Software Engineering (SBSE). While MSR formulates software engineering problems as data mining problems, SBSE reformulates SE problems as optimization problems and use meta-heuristic algorithms to solve them. Both MSR and SBSE share the common goal of providing insights to improve software engineering. The algorithms used in these two areas also have intrinsic relationships. Combining these two fields is useful for situations (a) which require learning from a large data source or (b) when optimizers need to know the lay of the land to find better solutions, faster.

Prof. Tobias Friedrich and I have employed the DSE concept in 2017 to investigate heuristic search approaches to extensively explore the design space of heuristics for certain optimisation problems [1, 2]. We distilled from the results new designs which our team was able to analyse theoretically, proving that the new approaches performed asymptotically better than state-of-the-art approaches.

So far, we have only investigated relatively simple single-objective problems, which can be analysed theoretically, such as linear functions and some submodular functions. What is missing is the connection to single- and multi-objective real-world problems. This is what this project aims to do. We will use automated algorithm configuration to optimise heuristics for classes of problems and to families of problem instances. We are hoping to achieve breakthroughs similar to the ones achieved last year.

1.2 Problems with Interconnected Components

In optimization research, problems with different characteristics are investigated. To find an appropriate algorithm for a practical problem often assumptions about characteristics are made, and then a suitable algorithm is chosen or designed. For instance, an optimization problem can have several components interacting with each other. Because of their interaction they form an interwoven system where interdependencies in the design and the objective space exist. An optimal solution for each component independently will in general not be a good solution for the interwoven optimization problem. In order to provide an academic interwoven optimization test problem, the Traveling Thief Problem (TTP) was proposed in 2013 [4] where two well-known subproblems, the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP), interact with each other. As in the TSP problem a so-called thief has to visit each city exactly once. In addition to just traveling, the thief can make profit during his tour by stealing items and putting them in the rented knapsack. However, the thief's traveling speed decreases depending on the current knapsack weight, which then increases the rent that the thief has to pay for the knapsack. The TTP's canonical formulation is a single-objective one, however, because the traveling time and profit represent solutions with different trade-offs, the problem is bi-objective in nature. To date, pretty much no research has considered this.

2 Non-Dominated Sorting Biased Random-Key Genetic Algorithm (NDS-BRKGA)

Figure 1 illustrates the overall procedure of our highly-configurable NDS-BRKGA. At first, we generate the initial population using efficient solvers for the subproblems independently. Afterward, we combine the optimal or near-optimal solutions for both subproblems and convert them to their genotype representation which results in the initial population. For the purpose of mating, the population is split into an elite population $P_e^{(t)}$ and non-elite population $P_{\bar{e}}^{(t)}$. The individuals for the next generations $P^{(t+1)}$ are a union of the elite population $P_e^{(t)}$ directly, the offspring of a biased crossover and mutant individuals. In case an individual violates the maximum capacity constraint, we execute a repair operation. Then, we convert each individual to its corresponding phenotype and evaluate it on the problem instance. In order to insert an explicit exploitation phase in our algorithm, we apply at some evolutionary cycles a local search procedure in some elite individuals. Finally, the survival selection is applied and if the termination criterion is not met, we increase the generation counter t by one and continue with the next generation. In the following, we describe the purpose of each of the design decisions we have made and explain what role it plays during a run of the algorithm.

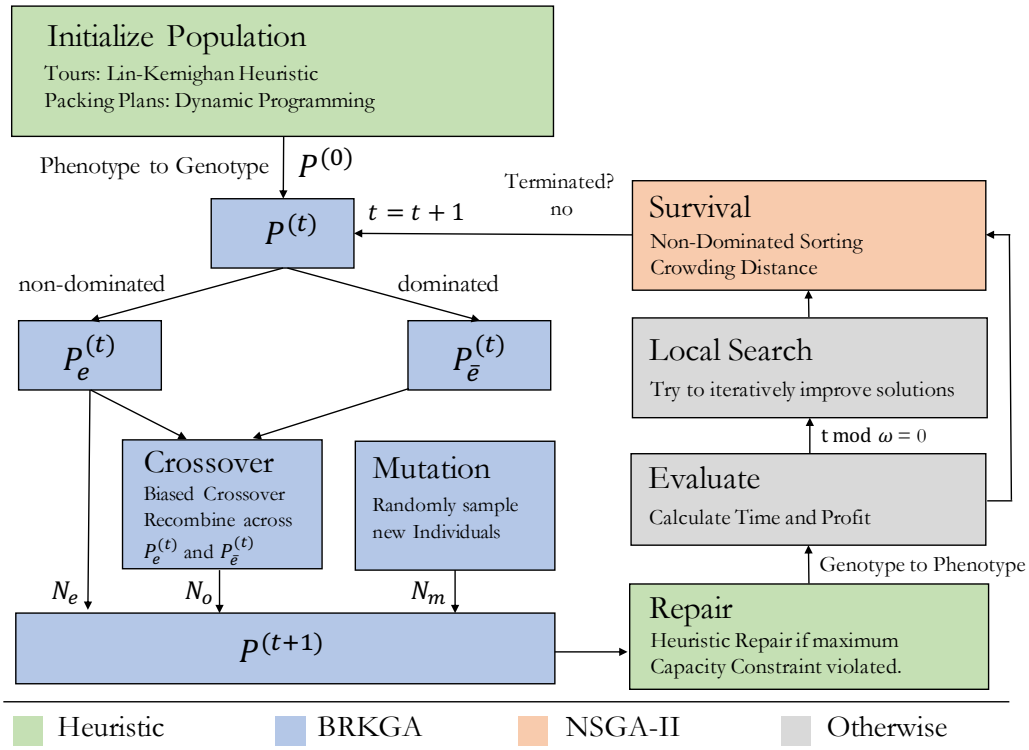


Figure 1: NDS-BRKGA: A customized genetic algorithm

3 Used Future SOC Lab resources

I have used the so-called 1000-core cluster of the HPI Future SOC Lab for the exhaustive hyper-parameter study. Running all 972 combinations (see table 1) on 9 instances, and performing 10 independent runs of 5 hours each consumed over 50 CPU years in the final experiments.

The parameters were population size N, elite population size N_e, mutant population size N_m, elite allele inheritance probability p_e, fraction $\alpha$of the initial population created from TSP and KP solvers, and the frequency ω for local search.

Table 1: Parameter values considered during the experiment

Parameter	Values
N	100, 200, 500, 1000
N_e	$0.3N$, $0.4N$, $0.5N$
N_m	$0.0N$, $0.1N$, $0.2N$
ρ_e	0.6, 0.7, 0.8
α	0.0, 0.1, 0.2
ω	10, 50, 100

4 Findings

Figure 2, we visualize the best parameter configurations at six different execution times. In each plot the best obtained parameter configuration regarding hypervolume is highlighted in red and parameter configurations up to 0.1 % worse than the best are highlighted in blue. Note that the importance of values of each parameter among the best parameter configurations is indicated by the intensity of the blue color once some parameter configurations share some parameter values. The following can be observed:

i. **More execution time, better results:** The number of parameter configurations that are capable of generating large hypervolume values increases as the execution time of our algorithm increases. This means that in some runs even though the hyperparameters were not set appropriately, the algorithm is still able to converge.

ii. **Importance of TSP and KP solvers:** It has influence on the overall performance of the algorithm if TSP and KP solvers are used for initialization which is determined by α. The best results are obtained if at least 10 % or 20 % percent of the initial solutions are biased towards those solutions found a TSP and KP solvers.

iii. **Trends when execution time increases**: We can see a trend as the execution time increases. Our method performs better with a large population, a large survival rate, a small or no explicit diversification through mutant individuals, a small influence of single-parent inheritance, an minor influence of a good initial population, and significant influence of local search procedure.

5 Next Steps

While the infrastructure allowed us to perform a great number of experiments, we have barely been able to scratch the surface. Additional experiments will be needed to further explore the data-driven design of our custom approach to this problem.

Among the next steps is the write-up of the preliminary findings in a scientific article – which will be submitted following HPI's review of the draft.

We plan to apply for an extension of the current project, as DSE requires to be driven by potent hardware. DSE is in its infancy, and it will become increasingly important – see the following claim from our inaugural DSE paper [3]:

> "Claim4: Data mining without optimization should be deprecated. Conclusions reached from an unoptimized data miner can be refuted, just by running the same tuned learner on the same data [35]. Since they can be so easily refuted, this community should stop publishing analytics papers that lack an optimization component."

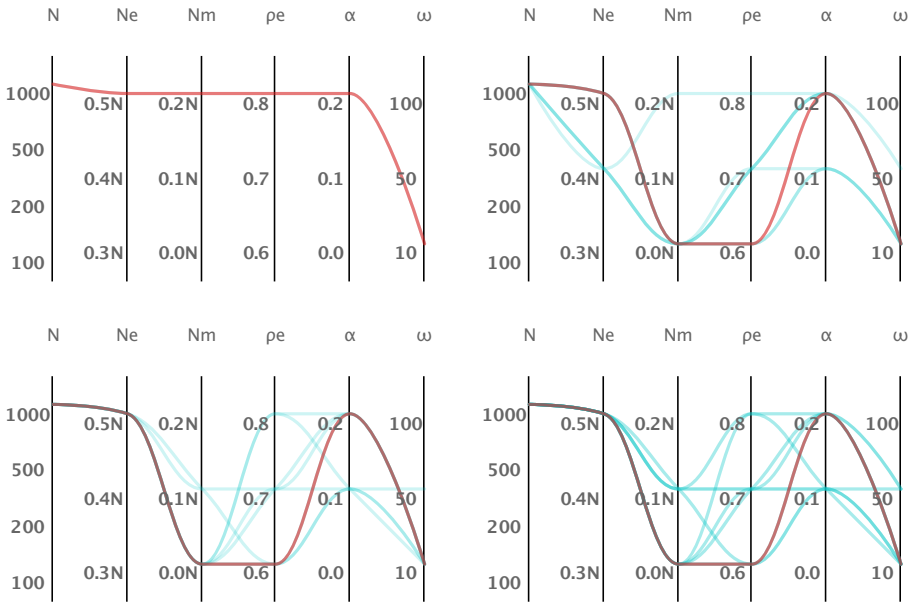


Figure 2: Best parameter configurations over all instances with varying execution times. From top left to bottom right: 600/1800/7200/18000 seconds.

Acknowledgments
Many thanks to the Hasso Plattner Institute and to the HPI Future SOC Lab team for making this research possible. I would also like to thank Jonatas B. C. Chagas and Julian Blank for their contributions to this research. This research has been partially supported by the ARC project DE160100850.

References

[1] T. Friedrich, A. Göbel, F. Quinzan, and M. Wagner. "Heavy-Tailed Mutation Operators in Single-Objective Combinatorial Optimization". In: *Parallel Problem Solving from Nature – PPSN XV*. Edited by A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley. Springer, 2018, pages 134–145.

[2] T. Friedrich, F. Quinzan, and M. Wagner. "Escaping Large Deceptive Basins of Attraction with Heavy-tailed Mutation Operators". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: ACM, 2018, pages 293–300. DOI: 10.1145/3205455.3205515.

[3] V. Nair, A. Agrawal, J. Chen, W. Fu, G. Mathew, T. Menzies, L. Minku, M. Wagner, and Z. Yu. "Data-driven Search-based Software Engineering". In: *Proceedings of the 15th Int. Conf. on Mining Software Repositories*. MSR '18. Gothenburg, Sweden: ACM, 2018, pages 341–352. DOI: 10.1145/3196398.3196442.

[4] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann. "A Comprehensive Benchmark Set and Heuristics for the Traveling Thief Problem". In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO '14. Vancouver, BC, Canada: ACM, 2014, pages 477–484. DOI: 10.1145/2576768.2598249.

How efficient is the use of In-Memory Database in Business Intelligence / BigData reporting for Blockchain based data?

Abdessatar Ben Ifa

University of Applied Sciences Ruhr West
abdessatar.ben-ifa@stud.hs-ruhrwest.de

Today's enterprises have large amounts of data that come from multiple sources which must be combined and processed efficiently. This data should also deliver in a Business Intelligence (BI) system for companies exactly the information they need in their current situation to make the right decisions. So, it is also possible to respond immediately to changes. In this case, even complex analyses and calculations must have a duration of only a few seconds instead of minutes or even hours. This type of system is used, for example, in the area of reporting and analysis in specialized areas such as sales, marketing, human resources, finance, purchasing or IT as well as corporate planning. However, the critical success factor is the performance of the system. Especially with large amounts of data, today's systems reach their limits quickly, which means that the required information is only available after a long waiting time. These waiting times can be unacceptable for users in times of cloud services and mobile devices. To close this gap, the access speed to the data must be improved as a major performance factor. Previously, this was limited in conventional database management systems using hard disk drives. Therefore, the use of In-Memory database that promise high speed is required. This Project evaluates Business Intelligence and BigData Reporting for Blockchain based data. The hybrid application, which combines the memory database and the traditional database, is also used and tested based on SAP HANA and MySQL.

1 Introduction

The rapidly growing data volume, optimized process times and the desire for reporting in real time are in conflict with today's way of keeping data for enterprise applications. The current standard for enterprise application data storage is a relational database system with disk space, where disk space is the bottleneck of the database and thus of the enterprise application. A optimization potential is possible due to the utilization of an In-Memory Database. Where the data is stored much faster and processor-related main memory of a server instead of in slow hard disk space. Technological advances in the field of processors (multi-core) and of main memory (capacity in the terabyte range per server) [1] have made it possible to rethink the approach of the classical relational databases. In previous research, In-Memory databases are mainly viewed from a technical perspective. And to narrow

this research gap, this work examines and analyzes the use of In-Memory Database in Business Intelligence and BigData reporting for Blockchain based data.

2 Project Idea

To examined the use of In-Memory database in practical operation, a preliminary application scenario has been selected. Thematically, this scenario should focus on complex queries in the field of Business Intelligence and should also work with a larger amount of data. It was a perfect solution to use the wavesBI as a data source, where it is based on Blockchain data with 16 tables and over 97 GB in size. And as (Bruce Lindsay, IBM) once said [2] : „Three things are important in the database world: performance, performance, and performance“. Some tests are used to determine how the database systems perform in some situations. The execution time of 5 complicated queries is intended to show the speed of the In-Memory database compared to other conventional databases. The two different databases technologies (SAP HANA and MySQL) was tested with 3 experiments (with and without indexes):

1. On the console (querying directly on the two DBS console).
2. Web-server (querying through a web-server).
3. Hybride-Modell (parallel querying over conventional database and In-Memory database).

3 Used Future SOC Lab resources (SAP HANA)

With the cooperation with the Hasso Plattner Institute, the Instance of In-Memory databases (SAP HANA) are available. Which enables to store a real data from the wavesBI database. And it make it possible to test the performance of the In- Memory database in Big Data and Business Intelligence report.

4 Results

4.1 Console Experiment

After that queries are automatically converted from SQL to OpenSQL with the own developed javascript software, it had to be executed against the databases sql and SAP HANA.

The experimental results of the various tests are shown in the Figure.1. It was carried out in the form of some tests making it possible to evaluate and compare the two types of traditional databases DBMS (MySQL) and In-Memory (SAP HANA)

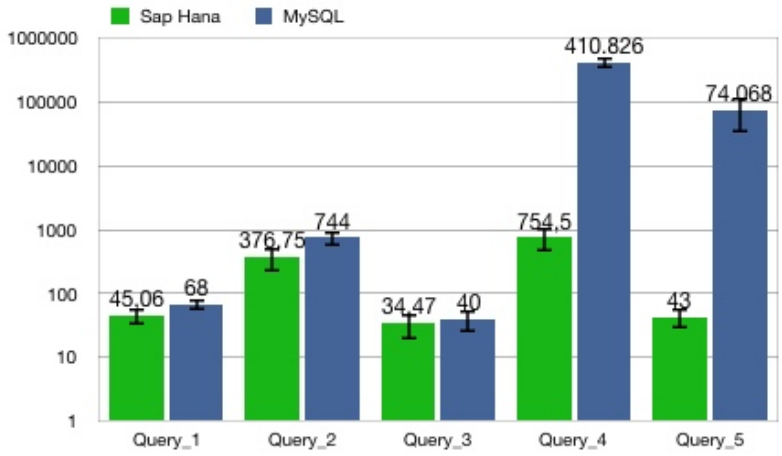


Figure 1: Query time Konsole(ms)

according to the execution time of the different careers and to maintain the same operating conditions to be executed each time. During the first test on the console, SAP HANA was very efficient by recording significant deviations from MySQL. In which the largest difference was recorded in the loading of Figure.1 with 43 ms for SAP HANA against (74068 ms) for MySQL. This poor performance of MySQL was due to SAP HANA database being designed to make complex selections of large volumes of data in a short time, unlike MySQL. Thereby it should be noted that, if the queries become more complex, the difference in response time between SAP HANA and MySQL becomes even greater than for simple queries such as Query-3 (34 ms for SAP HANA and 40 ms for MySQL).

4.2 Web-server experiment

The second experiment measured the required time to execute the queries across the two databases (MySQL and SAP HANA). And to achieve this goal, a web server (Node js) is implemented to get the elapsed time.

The different steps to follow are:

- The first one concerns the connection to the (SAP HANA or MySQL).
- The second concerns the execution of queries over the database concerning.
- The third concerns the response time.

Hence the performance of the databases was defined based on the speed of execution of operations. With the help of the Apache Benchmark. Each Web server was tested 100 times to keep only the average of the obtained response times.

In this test it highlighted that the best times were recorded by SAP HANA, who was much faster than MySQL in this phase. As in the first test on the console, SAP HANA became very efficient by recording the faster response time than MySQL. In

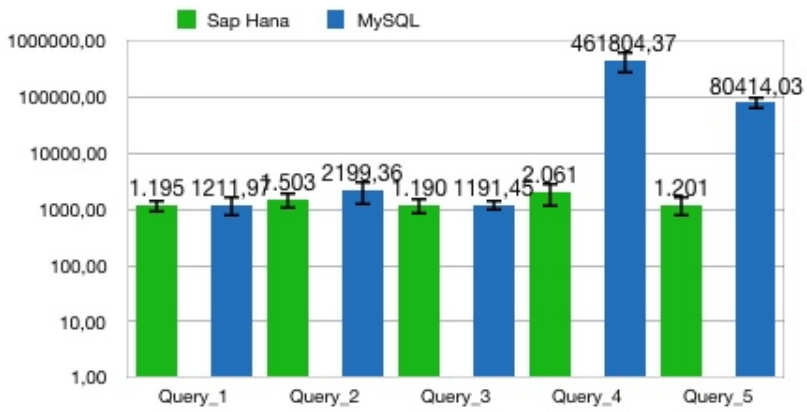


Figure 2: Query time web server(ms)

addition, the In-Memory database presented here by SAP HANA was faster than the traditional database technology (MySQL) overall queries.

4.3 The Hybrid Experiment

The third experiment was the so-called ”hybrid experiment”, which focuses on the implementation of the hybrid model by creating a new model that combines the in-memory database and a traditional database in a single system. Where the two databases used in this research are Sap Hana and MySQL.
And in the nonexistence of a finished solution, a Nodejs web-server is developed to accomplished this task. The idea was to save a small table in MySQL where as a big table would be in the In-Memory database. Then to query with join from two tables over the two databases and get the elapsed time. The different steps to follow are:

1. Connection to the databases (MySQL and SAP HANA).
2. Get all records from the concerned tables as JSON format.
3. Merge the results.
4. Query over the merged result.

After the execution of the Query-3 in the different experiment scenario. The following results are taken (74ms) for SAP HANA, (90.126 ms) for MySQL and (120 ms) for the hybrid experiment. It was not the best result yet, but it may be a topic for a future project to find an optimal solution for querying simultaneously over two different databases. This may allow a user to use the two different database technologies simultaneously. This solution can avoid too little storage space by the In-Memory database.

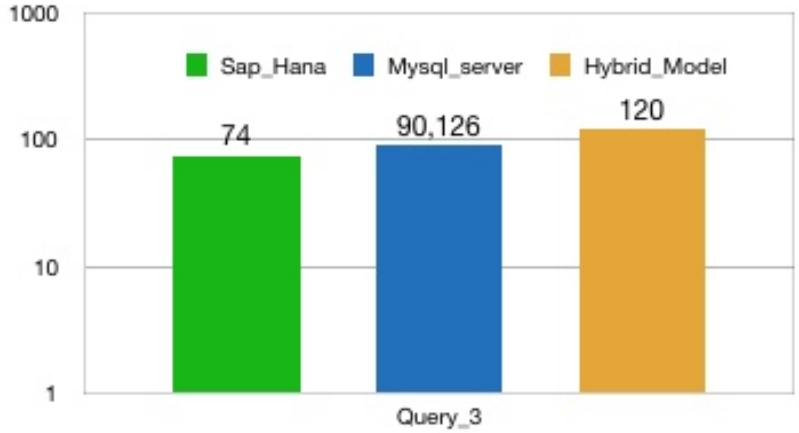


Figure 3: Query time web server(ms)

5 Conclusion and Future Work

In this work, an In-Memory database was examined and tested using the example of SAP HANA in the area of business intelligence reporting. Where the execution of the tests was subject to certain rules that were defined before the experiment and these are equally reliable. The main memory database results highlighted the high performance, especially with the very complicated queries, where the difference was always very large. The results of the data collected in this study recognizes the great need for fast and efficient processing of the constantly growing amount of data. So with its useful functions, the In-Memory database enables users to organize and analyze existing and emerging data sets in a meaningful way. Finally, it can be said that the advantages of the in-memory concept are even more important than the advantages of a traditional database, which can increase the performance of analyses and reporting. Another argument in favor of a switch to the in-memory concept is the falling price of RAM. Especially in the area of data warehousing, where it should bring a big advantage. And for the hybrid model, it could be improved in the near future. If a few tools are still developed like JSON Query and AlaSQL.

References

[1] T. Jüngling. "Datenvolumen verdoppelt sich alle zwei Jahre". In: *welt.de* (July 16, 2013). URL: https://www.welt.de/wirtschaft/webwelt/article118099520/Datenvolumen-verdoppelt-sich-alle-zwei-Jahre.html (last accessed 2019-01-01).

[2] W. Marianne. "Bruce Lindsay Speaks Out on System R". In: *ACM SIGMOD* (2005), pages 71–79.

Preliminary results on developing a glucose biomarker

Julia Sidorova[1], Lars Lundberg[1], Patrik Arlos[1], Ana Megia[2], and Joan Vendrell[2]

[1] Department of Computer Science
Blekinge Institute of Technology
{julia.a.sidorova}@gmail.com
{lars.lundberg}@bth.se
patrik.arlos@bth.se
[2] Endocrinology and Nutrition Service
University Hospital Joan XXIII
jvortega2002@gmail.com
ana.megia@gmail.com

Speech is an instantaneous, non-invasive, cost-effective biomarker in a wide spectrum of disorders and conditions, which makes it a preferable choice over currently used expensive, much-delayed and intrusive medical tests. Our previous work demonstrates the potential of a speech-based recognizer to approximate/predict of the glucose level. The problem of that study as well others reported in the literature was that it was not done on a statistically representative sample. A classical emotion recognition architecture is used as a predictive model. Currently, the project is at the data collection stage and we are using the IT resources of the HPI Future SOC Lab.

1 Introduction

A biomarker is "a characteristic that is objectively measured and evaluated as an indicator of normal biological processes, pathogenic processes, or pharmacologic responses to a therapeutic intervention" [1]. A speech-based biomarker [2] takes speech as an input and through it evaluates another variable of interest (for example, stress, physical load, and so on) at either a particular time moment or as a trend during months. There is big bulk of literature describing computational biomarkers that work on speech and share the same basic architecture and, to a large extent, the same implementation: stress recognition for physical load, for autism spectrum, for Parkinson's Disease, for Alzheimer's Disease, and intoxication. The proposed predictive model for glucose recognition is based on the same idea, which was initially meant for emotion recognition. The rest of the report is organized as follows. In section 2 the classical design of an emotion recognizer is reviewed. Section 3 provides a description of the data, which is being collected using HPI premises. Finally section 4 describes the future work.

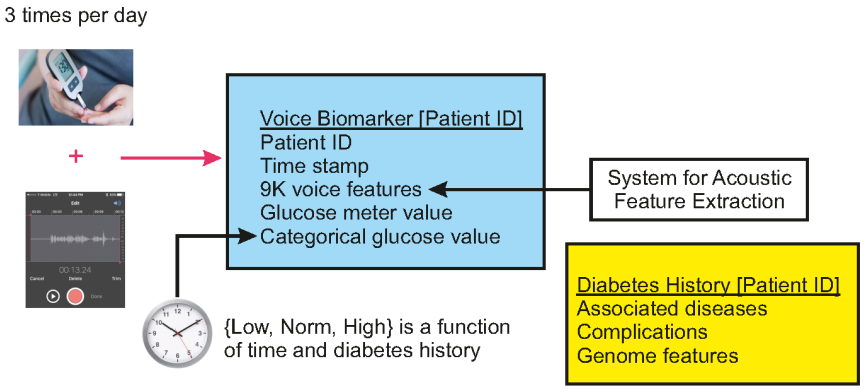


Figure 1: Database collection for diabetes voice biomarker

2 Classical Emotion Recognizer

A predictive model designed to recognise emotion (and other cognitive or physical conditions) from voice takes a speech sample as an input and outputs a class value for it. Typically, its architecture is based on the supervised machine learning approach. Variations are possible in the classical design with respect to preprocessing, features, and a classification function.

Step 0: Preprocessing options. The Z normalisation, i.e. all the features are set to mean 0 and sd 1, gives good results in cross-corpus recognition, and its purpose is to achieve speaker normalisation. (In this study, Step 0 has not been applied. Cross corpus recognition means that the speakers' IDs in train and test sets are different, i.e. speech samples by the speaker are either in the training set or in the test set. This is done to prevent the model from learning individual traits.)

Step 1: Feature Extraction. The state of the art acoustic feature set has been extracted from the speech signal: approximately 9K acoustic features with extracted with state of the art feature extraction software.

Step 2: Classification. Since there are many features, we have done a feature selection step, which implements correlation based feature subset selection (BestFirst and CfsSubsetEval in weka). The literature suggests that a classifier from the Support Vector Machine family is the most accurate method for classification in the experiments similar to ours, and we decided to use the SVM as implemented in weka.

Given that the classification task is new, and it had not been known whether the glucose level is detectable from voice, it was decided to build two predictive models: 1) for glucose >9 vs rest, and 2) for glucose <4 vs rest. The answers by the models can be connected into the final answer <4, norm, >9 via a probabilistic graph or in other ways.

3 Ongoing Data Collection

The ethical permissions were passed at Research Hospital Joan XXIII in Spain. The patients diagnosed with diabetes Type 1 were explained the purposes of this study and volunteered to participate, and have been collecting the data since August 2019. For now 20 patients have participated and 250 samples have been obtained. The program is thought as a continuous data collection. The recordings were made with their smartphones with highest quality available and sent over to the HPI premises for storage and further analysis. The participants were instructed to read the following two sentences from a prompt at a recording time:

1. El barco entro ayuer en el puerto. (The boat entered the harbour.)
2. Se siete atraido por esta chica. (He is attracted by the girl.)

Each recording is described with a time stamp, speaker identification, a glucose value measured with blood glucose meter. The measurement protocol was as follows:

1. Blood sugar measurement with a stitch was taken.
2. The participant entered a quiet room.
3. The recording of the prompted sentences were made.

Additionally medical history of the patients is available for the study. Currently one more hospital is waiting for the ethical permissions, and the data collection is planned to be extended.

4 Future work

The possibility to accurately detect blood glucose will be either confirmed or disproved on a statistically representative sample.

References

[1] V. O. Puntmann. "How-to guide on biomarkers: biomarker definitions, validation and applications with examples from cardiovascular disease". In: *Postgraduate Medical Journal* 85.1008 (Sept. 2009), pages 538–545. ISSN: 0032-5473. DOI: 10.1136/pgmj.2008.073759.

[2] J. Sidorova, S. Karlsson, O. Rosander, M. L. Berthier, and I. Moreno-Torres. "Towards disorder-independent automatic assessment of emotional competence in neurological patients with a classical emotion recognition system: application in foreign accent syndrome". In: *IEEE Transactions on Affective Computing* (2019). DOI: 10.1109/TAFFC.2019.2908365.

Benchmarking of Federated Learning

Comparison of three algorithms

Bjarne Pfitzner

Connected Healthcare Group
Digital Health Center, Hasso Plattner Institute for Digital Engineering
bjarne.pfitzner@hpi.de

The development of federated learning as a means to train machine learning models on a distributed and private dataset resulted in a multitude of algorithms for training such a system. Being able to choose the best algorithm for different situations is important, because exploring different methods is a time-consuming and computationally expensive task and should not be repeated by everyone who wants to implement federated learning. This work is a basis for a more extensive benchmarking of further federated learning algorithms.

1 Introduction

Federated learning is a privacy-preserving distributed machine learning paradigm between a server instance and multiple participants. The server brings some machine learning model it wants to train but lacks sufficient data for the learning process. The participants or clients on the other hand have some amount of training data, but might not have the expertise or amount of data to build a model themselves. Following the rules of federated learning, the model can be jointly trained while still preserving the privacy of each participant's data by not transferring it between any of them.

Since the publication of the first federated learning algorithm in 2016 [5], a number of researchers have investigated ways to optimise the training procedure. This report outlines two additional algorithms and presents a comparison of their effectiveness. The details of the considered algorithms are laid out in section 2.

2 Background

This section provides an overview over the considered federated learning algorithms.

2.1 Federated Averaging

The first instance of federated learning is given by federated averaging (`FedAvg`) [5]. The server communicates with K clients k, holding n_k data samples, which means there are $n = \sum_{k=1}^{K} n_k$ samples for training overall. During training the server selects

Algorithm 1: `FederatedAveraging (FedAvg)` [5]

1: **Server executes:**
2: Initialise θ^0
3: $m \leftarrow \max(C \times K, 1)$
4: **for** $t = 1$ to T **do**
5: $S^t \leftarrow$ (random set of m clients)
6: **for** each client $k \in S^t$ **do**
7: $\theta_k^t \leftarrow$ **ClientUpdate**(θ^{t-1})
8: **end for**
9: $\theta^t \leftarrow \sum_k \frac{n_k}{n} \theta_k^t$
10: **end for**
11:
12: **ClientUpdate**(θ): ▷ for client k
13: $\theta_k \leftarrow \theta$
14: **for** each local iteration **do**
15: **for** each batch b_k in client's split **do**
16: $\theta_k \leftarrow \theta_k - \eta \nabla L(b_k; \theta_k)$
17: **end for**
18: **end for**
19: **return** local model θ_k

a fraction C of clients and provides them with the current global model parameters θ^{t-1}. C is a hyperparameter that is defined by the server and can be optimised depending on the model complexity, connection and computation characteristics of clients. Then the clients train the model locally on their data and transmit the updates to model weights back to the server. There the updates are aggregated by computing the average, weighted by the number of training samples per client. After a number of global epochs, the overall model can be transmitted to all participants, making it available for predictions. Algorithm 1 shows this process as well.

2.2 LoAdaBoost

In cases where the data is not independent and identically distributed (non-IID), [2] claim that an adaptive boosting of the learning process, based on the loss, can make the training more robust and stable. In addition to the `FedAvg` algorithm, `LoAdaBoost` takes into account the median cross-entropy loss across clients, in order to boost the performance of weaker clients, that reach higher loss value for their local optimisation. Thus, each client has to return the loss $L(b_k; \theta_k)$ in line 19 of algorithm 1, and the server has to compute their median after line 9, as well as sending this median to all clients in line 7.

The median loss value is used by `LoAdaBoost` is that clients usually train only for $E/2$ local epochs. Only if the local loss after half the original epochs is still larger than the median loss across clients, then the current client continues training for

another $E/2 - trial + 1$ epochs (where *trial* denotes how often the local training was executed and tried to beat the median loss, starting with 1). The threshold for the total local epochs is set to $3E/2$, after which the current model parameters are sent to the server anyways.

This procedure is said to reduce the amount of local computations necessary to reach the same accuracy as in `FedAvg`.

2.3 Co-Learning

The idea for this collaborative learning approach (called `co-learning` [6]) is to increase local training efforts when the model updates stagnate, while also avoiding falling into local optima while optimising. The first part is done by increasing the number of local epochs over time. The amount of local epochs in global epoch t is given by

$$E^t = \begin{cases} E^0, & \text{if } t = 1, \\ 2 * E^{t-1}, & \text{if } t > 0 \ \& \frac{|\theta^t - \theta^{t-1}|}{|\theta^{t-1}|} \leq \epsilon, \\ E^{t-1}, & \text{if } t > 0 \ \& \frac{|\theta^t - \theta^{t-1}|}{|\theta^{t-1}|} > \epsilon. \end{cases} \tag{1}$$

After the server set the current local epochs, clients selected for participating in the global epoch are sent this new parameter together with the current model weights. Instead of (or even in addition to) a standard learning rate decay, `co-learning` employs a cyclical learning rate per client and global epoch. That means the learning rate of a client decays quite significantly during local training, but is reset the next time the client is picked by the server. Specifically the learning rate for local epoch e is given by

$$\eta^e = \eta^0 * \lambda^{e/E^t}. \tag{2}$$

Here η^0 stands for the learning rate at the beginning of the current global epoch t, which could be the same during the whole training, or decay over time.

Figure 1 shows a representative training process, where the local epochs were doubled in epochs 7 and 15. The accuracy curve shows a nice and stable learning process over time.

3 Implementation

The aforementioned algorithms were implemented in Python using Tensorflow 2.0 and Keras; the code is available on GitHub.[1] The client-server architecture of federated learning was simulated locally.

Table 1 shows the static training parameters. The learning rate is decayed in a time-based way, using the equation $\eta_t = \eta_0/(1 + \lambda t)$, where λ denotes the decay rate.

[1] https://github.com/BjarnePfitzner/FLBenchmarking/ (last accessed 2020-04-01).

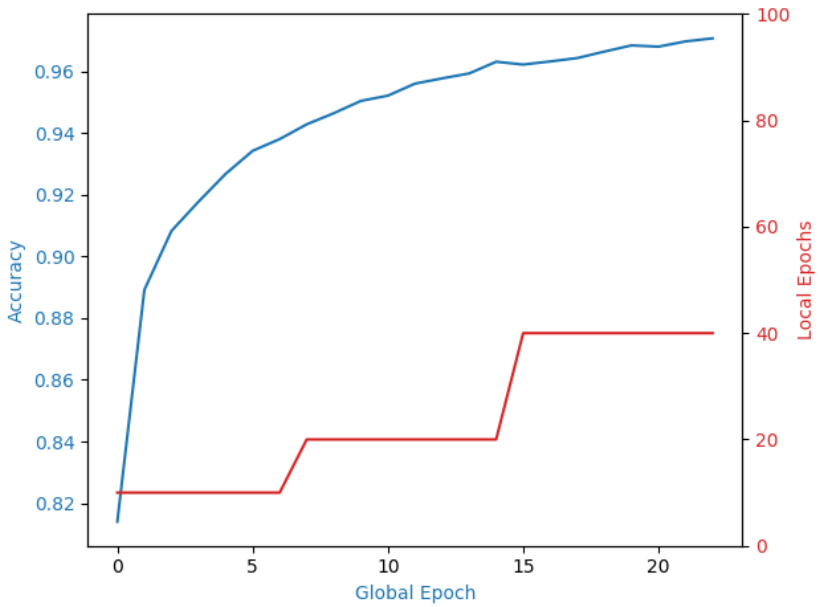


Figure 1: Model accuracy per global epoch, together with the number of local epochs selected by the `Co-Learning` algorithm. Results of a single run.

For Co-Learning the maximum amount of local epochs is set to 100 to avoid the possibility of infinite epoch doubling. The learning rate decay as given in table 1 stays in place, but additionally, the cyclical local learning rate decay is set to 0.25 as suggested by the authors of the original paper.

Table 1: Fixed training parameters

Parameter	Value
# Clients K	100
# Local samples n_k	600
Learning rate η	0.01
Learning rate decay λ	0.001
Convergence precision ϵ (see 2.3)	0.01

3.1 Dataset

The data included in this benchmarking is the common MNIST dataset [3] of handwritten digits. The samples are greyscale images of size 28x28 and were normalised to the range $[0, 1]$.

The data is distributed to the clients in an independent and identically distributed (IID) manner by randomly picking a subset for each client.

3.2 Model

The model investigated in this report is a Convolutional Neural Net (CNN) with two 5x5 convolutional layers with 32 and 64 channels respectively, followed by a 512-nodes fully connected hidden layer with ReLu activation and the output layer with 10 output nodes and softmax activation. I chose this model structure, because it has already been used in related literature [1, 4].

4 Evaluation

Models were trained until the global model reached an accuracy of 97 % or higher when evaluated on the held-out MNIST test set. The measure for algorithm efficiency is here the rounds of communication (RoC) required to achieve this accuracy. Experimentation has shown that average class-precision, -recall and -f1-measure do not give additional information, so they were omitted in this report.

As baseline a central model fed with the full 60.000 sample MNIST train set and using a batch size of 10, reached the required 97 % classification accuracy after a single pass over the data.

The results in table 2 show clearly that choosing an infinite batch size, i.e. considering the whole local dataset as one batch, is a bad choice and results in long convergence times. Moreover `LoAdaBoost` failed to converge to 97 % or higher accuracy within 1000 global epochs in four out of twelve experiments. Also for smaller batch sizes, it seems that this algorithm is inferior to the other two.

Threats to Validity

Due to limited time the different hyperparameter configurations have only been run once each, meaning the true metrics could vary a bit when averaged over e.g. 10 runs.

5 Future Work

Since this benchmark only scratches the surface of possible experimentation, I plan to continue investigating more settings and algorithms. The grid search employed in this work takes a long time, because it multiplies the number of possible hyperparameter values. Moreover, I would like to include more datasets, like CIFAR-10 a medical dataset or a synthetic one. This data could not only be distributed in an IID manner, but also in a biased and non-IID way. This is an interesting setting, because federated learning generally improves the performance of a model in the presence

Table 2: Results for different hyperparameter settings and algorithms. The yellow rows denote settings where the model convergence was very good (< 30). A batch size of ∞ means, that the whole local dataset as used as a single batch for local training. The percentages for `LoAdaBoost` show the accuracy the model achieved after 1000 global epochs.

Algorithm	C	B	E	RoC
`FedAvg`	0.1	∞	5	528
	0.1	∞	10	237
	0.1	10	5	12
	0.1	10	10	11
	0.5	∞	5	497
	0.5	∞	10	232
	0.5	10	5	11
	0.5	10	10	8
	1.0	∞	5	522
	1.0	∞	10	221
	1.0	10	5	12
	1.0	10	10	9
`Co-Learning`	0.1	∞	5	63
	0.1	∞	10	87
	0.1	10	5	13
	0.1	10	10	12
	0.5	∞	5	81
	0.5	∞	10	57
	0.5	10	5	22
	0.5	10	10	12
	1.0	∞	5	81
	1.0	∞	10	88
	1.0	10	5	18
	1.0	10	10	9
`LoAdaBoost`	0.1	∞	5	(90.96 %)
	0.1	∞	10	(94.26 %)
	0.1	10	5	134
	0.1	10	10	233
	0.5	∞	5	(96.60 %)
	0.5	∞	10	357
	0.5	10	5	21
	0.5	10	10	17
	1.0	∞	5	(96.83 %)
	1.0	∞	10	498
	1.0	10	5	27
	1.0	10	10	20

of non-IID data, but is still weaker than in the IID case. More specifically, each client receives only data from two classes, while all classes are still present in the same frequency (*2-class non-IID*). This is a very common setting used in federated learning research to evaluate an algorithm's performance in the presence of non-IID data [7].

6 Conclusion

This report showed an initial experimentation with three different federated learning algorithms.

In this limited experimentation, there was not a big difference between the convergence speed of `FedAvg` and `Co-Learning`. `LoAdaBoost` on the other hand performed significantly worse. I expect that the introduction of non-IID data will change this outcome and favour the more sophisticated `Co-Learning` and `LoAdaBoost` algorithms over `FedAvg`.

References

[1] Y. Chen, X. Sun, and Y. Jin. "Communication-Efficient Federated Deep Learning With Layerwise Asynchronous Model Update and Temporally Weighted Aggregation". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019). DOI: 10.1109/TNNLS.2019.2953131. arXiv: 1903.07424. In press.

[2] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu. "LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data". In: *CoRR* abs/1811.12629 (2018). arXiv: 1811.12629v2.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pages 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Edited by A. Singh and J. Zhu. Volume 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pages 1273–1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html (last accessed 2019-01-01).

[5] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas. "Federated Learning of Deep Networks using Model Averaging". In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629v3.

[6] K. Xu, H. Mi, D. Feng, H. Wang, C. Chen, Z. Zheng, and X. Lan. "Collaborative Deep Learning Across Multiple Data Centers". In: *CoRR* abs/1810.06877 (2018). arXiv: 1810.06877.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. "Federated Learning with Non-IID Data". In: *CoRR* abs/1806.00582 (2018). arXiv: 1806.00582.

Energy Efficiency, Virtualization and Performance
Third (WEEVILT)

Carlos Juiz and Belen Bermejo

Computer Science Department
University of the Balearic Islands, Spain
{cjuiz,belen.bermejo}@uib.es

1 Project idea

The prevailing international scientific opinion on climate change is that human activities resulted in substantial global warming from mid-20^{th} century, and that continued growth in greenhouse gas concentrations, caused by human-induced emissions, is mainly (direct or indirectly) due to energy consumption. The sector where energy consumption has grown tremendously is IT (Information Technologies). On one hand, datacentres that host cloud services are becoming huge warehouses with lot of servers, additional electronic equipment and complex cooling systems. These computers and telecommunications networks are today primarily responsible for electronical energy consumption caused by datacentres, which maintain the quality of Internet service in operation. Giving more capacity to these datacentres usually means more cloud servers and consequently additional more equipment and cooling are needed, too. On the other hand, the increasing number of users, especially due to the popularization of cloud services, produced continuously capacity problems at datacentres due to performance requirements [2, 3]. Thus, a new challenge directly related to this research area emerges: the energy efficiency of cloud servers. WEEVILS project is aimed by this combination of these topics. Our main project objectives are:

1. The characterization of the performance and energy consumption from consolidated servers through benchmarking and monitoring techniques.
2. Modelling the energy consumption patterns and performance of consolidated servers. We shall infer models of such energy, performance and virtualization, together.
3. Performing several comparative studies of benchmarking between physical and virtual servers.
4. The validation of our performance-oriented previous findings and energy efficiency virtualization models through experimentations in real datacenters, as HPI Future SOC Lab.

The WEEVILT project is defined as the extension of the previous one (WEEVILS) developed using the HPI Future SOC Lab infrastructure (RX600S5-1 server).

Then, we attempt to go in depth in CPU-overhead sources when consolidating virtual machines. Specifically, we study how the behaviour of OV_v and OV_c under different hypervisors is, as we state in the previous report as next steps (WEEVILS project).

2 Used Future SOC Lab resources

In order to answer the research question, we design a methodology based in the following stages:

1. To characterize the performance of physical servers through benchmarking and monitoring techniques [4].
2. Identify the sources of virtual machine consolidation overhead.
3. Measuring and quantifying the virtual machine consolidation overhead.
4. Discussing and analysing the results.

The exposed methodology was performed in the HPI Future SOC Lab infrastructure, specifically in the rx600s5-1 server which hardware has 48 CPUs and 1024 GB of RAM memory. In table 1 we can see the actions developed in each stage and the correspondent outcome.

Table 1: Actions developed in each stage and its outcomes

Stage	Action	Outcome
Stage 1	Characterizing the performance of PM through benchmarking and monitoring techniques	PM's performance when consolidating virtual machines
Stage 2	Identification of overhead sources	Description of virtual machine consolidation overheads
Stage 3	Measuring and quantifying the virtual machine consolidation overhead in HPI infrastructure and UIB's servers	Overheads' quantification
Stage 4	Discussion and results' analysis	Comparison between UIB and HPI servers in terms of overheads

3 Findings

In previous section we explain the stages we perform to answer the research question, as well as the developed actions and its outcomes. We define [1] the virtual machine consolidation overhead as the extra work that the system has to perform in order to manage the consolidated virtual machines. As a consequence, we identified two sources of overhead. The former is coming from the fact of having an hypervisor (OV_v), and the later is from the fact of having more than one allocated virtual machine (OV_c). Moreover, depending on the hypervisor implementation (Type-I, Type-II or container-based), the number of consolidated virtual instances and the exeuted workload, the values of OV_v and OV_c could be considerably different.

In figure 1,figure 2, and figure 3 we depict the graphical representation of the overheads' sources for the HPI server,that is,type-I, type-II and container-based hypervisor (respectively). We can observe that when the virtual machines are consolidated by type-I hypervisor (see Fig. 1) the percentage of the response time dedicated to perform useful work is higher than when we consolidate through type-II (see Fig. 2) or container-based (see Fig. 3) hypervisor. This fact is due to the proximity of the type-I hypervisor with the physical hardware.

Regarding the OV_v, we can see that this value is higher when consolidating with type-II hypervisor than type-I or container-based implementations. This fact is due to the type-II deployment. Since it is deployed on an operating system, the proximity with the physical hardware is higher than the type-I implementation. Besides, the value of OV_c is minimum when consolidating with type-II hypervisor. The contrary occurs for type-I and container-based hypervisor implementation. In the case of OV_c, its value depends on the number of consolidated instances.

In addition, we can observe that the consolidation degree varies for any hypervisor implementation. When consolidating with type-I hypervisor the number of consolidated virtual machined is 15, 9 for type-II hypervisor and 9 when we consolidate containers instead of virtual machines. This fact is due to the implementation of virtual machines. A container is considered as a system process, the same as type-II virtual machines. Then, the operating system needs more resources to manage them. Nevertheless, a type-I virtual machine is not considered as a system process, and it not need and operating system to be deployed.

3.1 Publications

The use of the HPI infrastructure collaborates to develop the following research works:

- B. Bermejo, C. Juiz, and C Guerrero. "On the Linearity of Performance and Energy at Virtual Machine Consolidation: the CiS2 Index for CPU Workload in Server Saturation". In: Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications. Exeter, UK, 2018, pages 928– 933.

- B. Bermejo, C. Juiz, and C. Guerrero. "Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance". In: The Journal of Supercomputing (2018). ISSN: 1573-0484. https://doi.org/10.1007/s11227-018-2613-1.
- B. Bermejo, C. Juiz, and N. Thomas, "On the virtualization overhead and energy consumption in consolidated servers." in UK- Performance Engineering Workshop (UKPEW), 2018.
- B. Bermejo and C. Juiz. "Virtual machine consolidation: a systematic review of its overhead influencing factors". In: The Journal of Supercomputing (2019). *Accepted and pending of publication.*

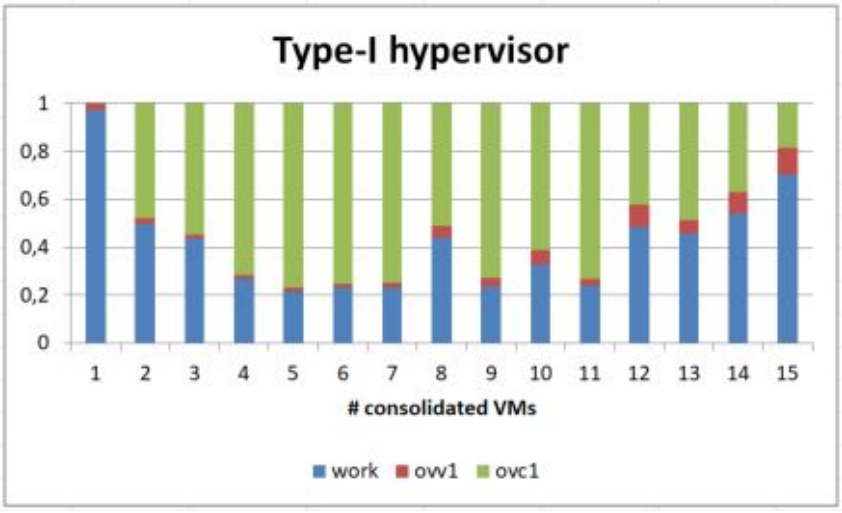


Figure 1: Descomposed response time for Type-I hypervisor

4 Next steps

In this project we attempt to answer the research question: "How to determine the performance degradation of physical servers due to Virtual Machine Consolidation?". For that, we design a methodology based on monitoring and benchmarking techniques and we apply it to our servers and then, we extend the experiment to HPI infrastructure.

The HPI allows us to extend the work and also to obtain more accurate results and conclusions. Due to that, there are very interesting research questions that we would like to answer with the support to the HPI infrastructure. The question is: "Could be generalize the behaviour of the virtual machine consolidation overhead?"

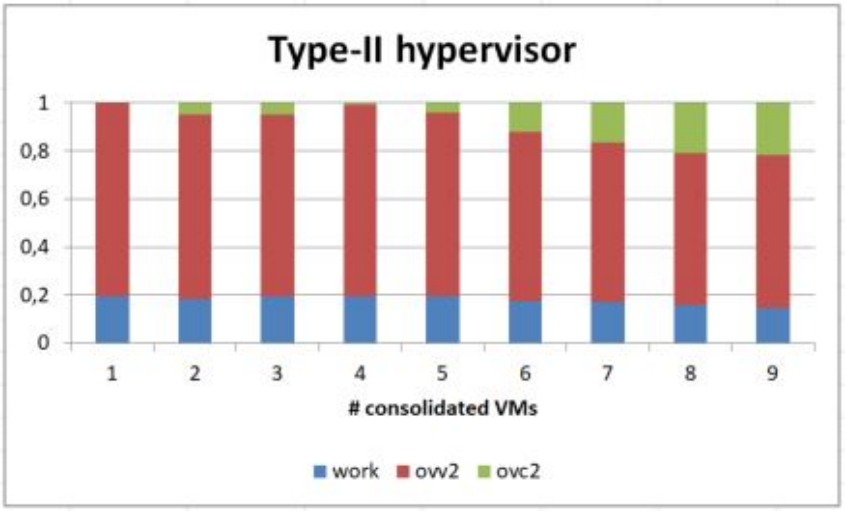


Figure 2: Descomposed response time for Type-II hypervisor

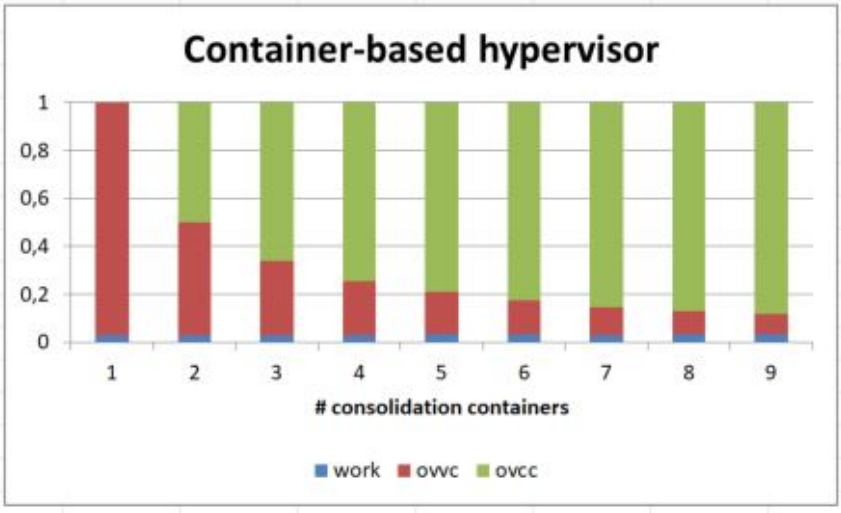


Figure 3: Descomposed response time for container-based hypervisor

In order to answer the proposed question, we will apply for a new project in the Hasso Plattner Institute in order to use the HPI Future SOC Lab's IT infrastructure.

References

[1] B. Bermejo, C. Juiz, and N. Thomas. "On the virtualization overhead and energy consumption in consolidated servers". In: *UK- Performance Engineering Workshop (UKPEW)*. Newcastle upon Tyne, UK, 2018.

[2] B. Bermejo, S. Filiposka, C. Juiz, B. Gómez, and C. Guerrero. "Improving the Energy Efficiency in Cloud Computing Data Centres Through Resource Allocation Techniques". In: *Research Advances in Cloud Computing*. Edited by S. Chaudhary, G. Somani, and R. Buyya. Singapore: Springer Singapore, 2017, pages 211–236. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_9.

[3] R. Buyya, C. Vecchiola, and S. T. Selvi. *Mastering Cloud Computing: Foundations and Applications Programming*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 978-0-12-409539-7.

[4] X. Molero, C. Juiz, and M. Rodeño. *Evaluación y Modelado del Rendimiento del os Sistemas Informáticos*. Pearson, 2004. ISBN: 978-84-205-5780-9.

The usage of State-Of-Art Neural Networks in the classification problems

The usage of XDeepFM for Musical Genre Classification

Joao Sauer and Leandro dos Santos Coelh

Universidade Federal do Paraná
Electrical Engineering Sector
joao.sauer@gmail.com, lscoelho2009@gmail.com

Nowadays, the musical genre can be very ample and in some cases, creates discussions about what a person would consider it as only one particular genre. This kind of classification makes machine learning algorithms struggle to analyse it correctly. However, some new technologies like Generative Algorithm Networks(GANs) or XDeepFM could be manipulated and used as a classification tool to try to solve problem. Helping to identify correctly a particular kind of music focusing only in identifying the most important genre of a music.

1 Introduction

The musical field has different information that can be extracted and an interesting field to be analysed [4]. However, its features are complex for a classification tool as one feature can be common between two or more genres. Different approaches have being used, such as the usage of sentiment analyses of the lyrics [12], the usage of a Convolutional Neural Network(CNN) [14], Recurrent neural networks (RNN) [15] or Long short term memory(LSTM) [5], most of this approaches also made a pre-selection of which fields should be used, using methods such as correlation matrix and finally normalised and grouped to be ready to be sent to the training methods. In this new phase of work, we suggest some modifications in the GAN and XDeepFM to allow it to be used as a classification tools, comparing its results with Random Forest or K-Means and identify the main musical genre from a song using a confusion matrix to verify the true correlation of the results.

Contributions

The contributions expected from this work is the usage of Generative adversarial network (GAN) [6] and eXtreme Factorial Machine(XDeepFM) [11] to be used as classification tools for the identification of the main musical genre from a song and compare the results with some classical methods, such as K-Means [3], T-SNE [13], SVM [2] and Random Forest [7]. This research combines two musical datasets The Million Song Dataset (MSD) [1] and Last.fm dataset [10]. The MSD contains features

extracted from one million songs. The Last.fm dataset gives the genre of the music based on the opinion from humans.

2 Context

Music genre classification have already being studied in the past using different approaches. One of the approaches uses sentiment analyses from the lyrics of the musics [4]. Another approach to classify the musical genre is the usage of random forest [8] or K-Means [9]. The results from all these studies proved it is possible to use an algorithm to solve this issue. However, there are some newer approaches that could be used on these datasets and it opens the opportunity to verify the quality of them against these datasets. Also, these datasets normally do not contain the audio of the music, just information about the songs and their metadata, which makes even more difficult for the classification.

3 Problem

The musical genre classification is a complex problem, once that different individuals can have distinct opinions about the same music. An individual can consider a music as *Pop-Rock* while another would consider it as only *Rock*. However, in most cases the chosen genres are co-related, which allows the usage of one common genre to be the primary. Another difficulty is the similarity between the features identified in the dataset. These similarities can mislead the correct classification of a song.

Specific Problem

In the MSD, there is no audio to be analysed. This research uses only the information provided by the dataset. One key piece of information is missing from the MSD: the genre. This information is provided by the last.FM dataset. Together, both datasets generate a new dataset that can be analysed. The merging of these datasets provides extra information from a single song such as its beats, bars, tatums, danceability, loudness and many other information.

4 Solution

Machine Learning can be used to solve this problem. It is possible to train the system to learn to identify a particular genre from a song. There are several approaches that can be used such as GAN, XDeepFM, random forest, K-Means. This research applies some of them, makes a comparison between them and suggests a new approach. This research using GAN tries to verify if an entry from the dataset is part or not of a particular group of individuals. This research modifies the original GAN with

clusterization of data. As a classification tool, this research uses K-Means. The other research, that involves a new proposal method, called XDeepFM, uses Compressed Interaction Network(CIN) with Deep Neural Networks (DNN) in one unique Model which allows it to learn certain bounded-degree feature interactions explicitly and at the same time it can learn arbitrary low and high order feature interactions implicitly.

Specific Solution

The GAN is an unsupervised learning method for neural networks. GAN normally is used to generate and analyse clusters of data. In this research the method was changed to support classification. On the other hand, xDeepFM is a Factorial Machine, that requires some modification in the dataset to also support this new approach. With that, the comparative between the classical models and the new ones can be set.

5 Implementation

In this phase, the docker image was created with the scikit and Jupyter Notebooks, together with Tensorflow. The following activities have been performed in the docker image:

- Dataset Manipulation;
- Usage of K-Means;
- Usage of Random Forest;
- Usage of GAN;
- Productivity of XDeepFM examples.

A big part o this phase was the analyse of the dataset, which contains too many information that can create bias that the models are not expecting. For example, some musics doesn't contain some of the fields, but others contains values that are related to other fields, creating an important correlation between them. So, most of the work is still in the analyses of the fields.

5.1 Dataset Manipulation

The MSD contains a million files separated in folders by the music name. Each file is a .h5 that contains all the information related each music. Most of the them contains all the possible information. The first step was read all files with all possible valid features in a unique file that could be used in Jupyter Notebook. In addition, the original files does not contain the musical genre. This information is in a separate dataset. The key between both datasets is an ID. As a first test, only the `beats_confidence` of the files was used to try to classify the songs in the different possible genres. In a second stage, more fields were analysed and used:

- bars start
- beats start
- danceability
- duration
- end of fade
- energy
- loudness
- start of fade out
- tatums start
- tempo
- time signature

Most of them have the confidence field, which also were used to confirm how useful can the field be.

5.2 K-Means

K-Means splits the dataset into genres and then map the different clusters with the most common genre found to confirm that the group found was an specific genre. Because it's an unsupervised learning, this could give the ability to give multiples genres, depending on the distance of the centroids found.

5.3 Random Forest

Random Forest, as the name suggest, is a tree creator that tries to use the features found as parameters to decide were and how deep the entry should go in the tree. With the training dataset, the tree is created and then a new test entry is passed. It decides which node in the tree belongs and returns its result. This is a supervised method and because of this, the results are unique. Together with Random Forest, two other methods, from the same package, were used: SVM and Kernel SVM. The expectation would be to use them as a comparative for the Random Forest method.

5.4 GAN

GAN is an unsupervised learning method for neural networks. The model consists of two networks. A generator that generates the data from a noise vector, and a discriminator that discriminates between a generated data and a real data. The interesting part is that once the network is trained it can generate the data from random noise that can be used to identify the correct genre for that particular data. For this phase, a very simple approach was used, using a random noise to generate the fake data. However, its possible to notice that the results show promising.

5.5 XDeepFM

xDeepFM is a Factorial Machine, that is quite recent and is showing good results. As suggested by the Professor, it would an important addition to the recent research and an analyse of its usage start to begun. For this phase, it was only used an example from github: https://github.com/shenweichen/DeepCTR (last accessed 2019-01-01) to verify how can be implemented for the dataset requested.

6 Evaluation

Because the project has just started the evaluation was not yet used. It was implemented using the default options from scikit-learn, as showed in listing 1.

Listing 1: Code used for evaluation

```
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
print("f1_score:",metrics.f1_score(Y_test, Y_pred))
print("recall_score:",metrics.recall_score(Y_test, Y_pred))
print("precision_score:",metrics.precision_score(Y_test, Y_pred))
```

The results can be looked at table 1. The GAN network was not evaluated at this stage of the research.

Table 1: Results Evaluation

	GAN	K-Means	Random Forest	SVC	Kernel SVC
accuracy	0.684	0.064	0.629	0.458	0.639
f1_score	0.682	0.064	0.629	0.458	0.639
recall_score	0.684	0.064	0.629	0.458	0.639
precision_score	0.688	0.064	0.629	0.458	0.639

Another important point to be noticed was the usage of the confusion matrix against the random forest tests. This can be seen in figure 1 below:

As explained, the new model approach XDeepFM, is not yet using the same dataset, so is not showed here. The next phase is focusing in doing the necessary changes on the code to support it.

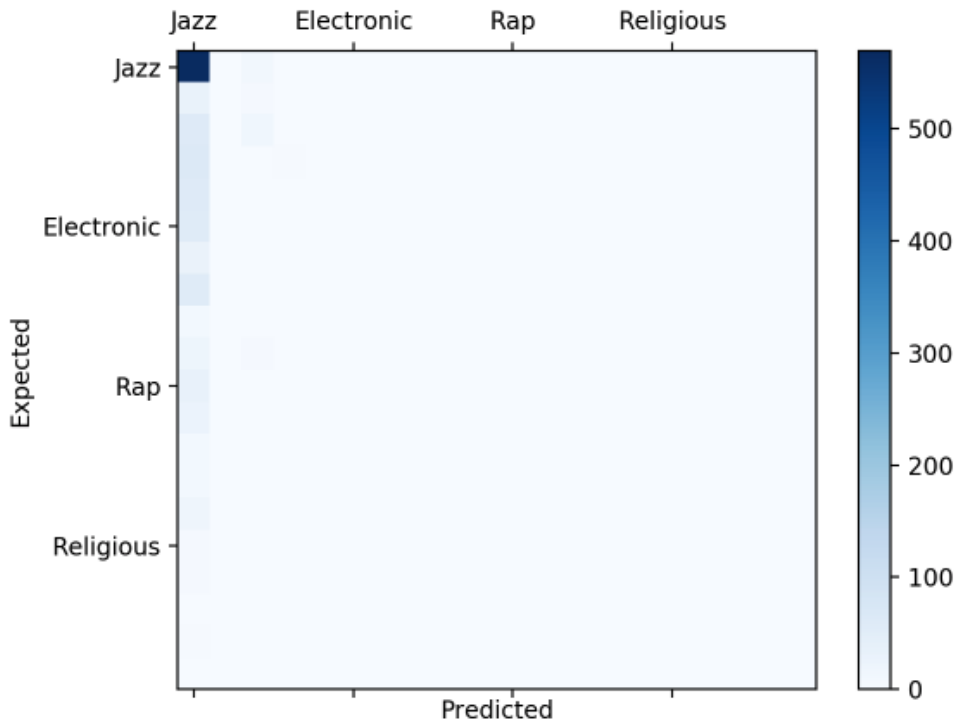


Figure 1: Confusion Matrix for Random Forest

7 Conclusion

As explained before, this work has just started. The development is still a work in progress. There are several work to be done, as showed by the results obtained so far. The dataset features are not the best ones to be used right now and its necessary to have a better approach of selecting them. On the other hand, it's possible to say that the GAN can be changed to fit the proposed work.

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[2] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pages 273–297.

[3] E. Forgey. "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification". In: *Biometrics* 21.3 (1965), pages 768–769.

[4] D. Gärtner. "Tempo Detection of Urban Music Using Tatum Grid Non Negative Matrix Factorization". In: *ISMIR*. 2013.

[5] D. Ghosal and M. H. Kolekar. "Music Genre Recognition Using Deep Neural Networks and Transfer Learning". In: *Proc. Interspeech 2018*. 2018, pages 2087–2091. DOI: 10.21437/Interspeech.2018-2045.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pages 2672–2680.

[7] T. K. Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Volume 1. IEEE. 1995, pages 278–282.

[8] X. Jin and R. Bie. "Random Forest and PCA for Self-Organizing Maps based Automatic Music Genre Discrimination." In: *DMIN*. 2006, pages 414–417.

[9] D. Kim, K.-s. Kim, K.-H. Park, J.-H. Lee, and K. M. Lee. "A music recommendation system with a dynamic K-means clustering algorithm". In: *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE. 2007, pages 399–403.

[10] *Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset*. 2011. URL: http://labrosa.ee.columbia.edu/millionsong/lastfm (last accessed 2019-02-07).

[11] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun. "xdeepfm: Combining explicit and implicit feature interactions for recommender systems". In: *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pages 1754–1763.

[12] D. Liang, H. Gu, and B. T. O'Connor. *Music Genre Classification with the Million Song Dataset*. Technical report 15-826 Final Report. Carnegie Mellon University, Dec. 3, 2011.

[13] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pages 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html (last accessed 2019-01-01).

[14] J. Pons and X. Serra. "musicnn: Pre-trained convolutional neural networks for music audio tagging". In: *Demo at ISMIR 2019*. 2019. arXiv: 1909.06654 [cs.SD].

[15] J. Yang. "Music Genre Classification With Neural Networks: An Examination Of Several Impactful Variables". In: *Computer Science Honors Theses* (2018). URL: https://digitalcommons.trinity.edu/compsci_honors/44 (last accessed 2019-01-01).

Exploring Game-Theoretic Formation of Realistic Networks

Tobias Friedrich, Pascal Lenzner, and Christopher Weyand

Algorithm Engineering Group
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

Many real world networks from different domains share the same structural properties. So far, there only exist models which reproduce these properties via a random process and thus have only limited explanatory power. In contrast to this, we have developed an agent-based game-theoretic model which promises a better explanation of the structure of real world networks. Our new model was investigated in computationally demanding large-scale experiments performed on the hardware of the HPI Future SOC Lab.

1 Introduction

Complex networks from the Internet to various (online) social networks have a huge impact on our lives and it is thus an important research challenge to understand these networks and the forces that shape them. The emergence of the Internet has kindled the interdisciplinary field of Network Science [3], which is devoted to analyzing and understanding real-world networks.

Extensive research, e.g. [1, 3, 4, 6, 13, 16, 17], on real world networks from many different domains like communication networks, social networks, protein-protein interaction networks, neural networks, etc. has revealed the astonishing fact that most of these networks share the following basic properties:

- *Small-world property:* The diameter and average distances in these networks are logarithmic in the number of nodes or even smaller.
- *Clustering:* Two nodes which are both adjacent to a third node have a high probability of being neighbors themselves, i.e. real networks contain an abundance of triangles and small cliques.
- *Power-law degree distribution:* In real networks the probability that a node has degree k is proportional to $k^{-\beta}$, for some constant $2 \leq \beta \leq 5$. That is, the degree distribution follows a power-law. Such networks are called *scale-free networks*.

The phenomenon that real world networks from different domains are very similar calls for a scientific explanation, i.e. formal models which generate networks with the above properties from very simple rules.

Many such models have been proposed, most prominently the preferential attachment model [4], the Chung-Lu random graph model [8], hyperbolic random graphs [12, 14] and geometric inhomogenous random graphs [5]. However, all these

models describe a purely random process which eventually outputs a network having realistic properties. On the one hand, this is desirable for sampling such networks, e.g. for testing algorithms on them, but on the other hand having a purely random process yields only a limited explanation of the structure of real world networks. Most real world networks evolved over time by the interaction of various rational agents. In case of the Internet the selfish agents correspond to the Internet Service Providers which control the Autonomous Systems, in case of social networks, the agents are people or companies who choose carefully with whom to connect. Thus, a model with higher explanatory power should consider rational selfish agents which use and modify the network to their advantage. Such models are at the core of the young field of Algorithmic Game Theory [18, 19].

2 Networks via Game Theory

In game-theoretic models for network formation selfish agents are associated to nodes of a network. Each agent chooses as strategy any subset of other agents to form a link to. The union of all links which are chosen by some player then determines the links of the created network.

The individual goal of each agent is modeled via a cost function, which typically consists of costs for creating links and of a service cost term, which measures the perceived quality of the created network for the individual agent, e.g. the service cost could be the sum of distances to all other agents [11] or just the number of reachable agents [2].

Any assignment of strategies to agents is considered an outcome of the game. Among all those outcomes the so-called equilibria are particularly interesting. In an equilibrium no agent wants to change her current strategy, given that all other players' strategies are fixed, i.e. no agent can reduce her costs in the current situation by forming another set of links. Analyzing the structure of such equilibrium networks then ideally yields insights into why real world networks exhibit the mentioned properties.

So far, such game-theoretic approaches can explain the small-world property, that is, it was proven that the diameter of all equilibrium networks is small [10]. However, to the best of our knowledge, no known game-theoretic model can explain the emergence of clustering and a power-law degree distribution. Thus, it is still an open problem to find and validate such a model.

3 Aims of the Project

Building on our previous work [7, 9, 15], we have developed a new game-theoretic model, called *strategic network augmentation*, which promises to solve the open problem. Initial experiments performed on the Future SOC Lab compute cluster [20, 21] revealed that the obtained equilibrium networks from our new model have the small-

world property, show significant clustering and the node degree distribution seems to be governed by a power-law. Moreover, our experiments established that certain minimum number of nodes is needed to reliably generate the desired properties. It turned out that experiments with $n = 1000$ nodes are rather inconclusive but that experiments with $n \geq 10000$ nodes yield valuable insights. This raised the question of reinvestigating the parameter space of the model for $n \geq 10000$ nodes.

The focus of this project was to reinvestigate whether the obtained power-law exponent and the local average clustering coefficient can also be adjusted for high number of nodes, i.e., for $n \geq 10000$. In a previous project [20] which focused on the case $n = 1000$ we established that the power-law exponent and the local average clustering coefficient correlate with the edge-price factor of the model. For higher edge-prices the obtained values where significantly lower, i.e., the edge-price factor is a means of adjusting the obtained network characteristics. Sophisticated algorithm engineering to improve the performance of our simulator [21] in combination with the high computing power of the HPI Future SOC Lab allowed us to check whether the results scale to large numbers of nodes.

4 Used Future SOC Lab Resources

The experiments were run on the high-performance cluster of the HPI Future SOC Lab. The cluster consisted of 22 nodes with 80 cores each and 1TB of memory each.[1] The experiments were run via the slurm job scheduler on all nodes in parallel.

In total we generated over 10000 networks with 1000 to 100000 nodes and extracted various properties along the way to monitor the process. Later, we analyzed the generated networks using various metrics.

5 Findings

We had some expected findings and some rather surprising findings. First of all, we could demonstrate that also for a high number of network nodes the network characteristics like the power-law exponent and the average local clustering coefficient can be influenced by adjusting the edge-price parameter α (See figure 1).

However, the experiments revealed that the influence of the edge-price factor for one of the considered models that included some form of locality is much weaker than expected (see figure 2) and, quite surprisingly, the observed power-law exponents seems to decrease for increasing edge-prices.

Another interesting finding for the model with locality was that the number of edges and the average node degree seems to be invariant to changes in the edge-price factor (see figure 3(left)). This is highly surprising since we expected much sparser

[1] Further hardware specifics can be found here: https://hpi.de/en/research/future-soc-lab/equipment.html (last accessed 2020-04-01)

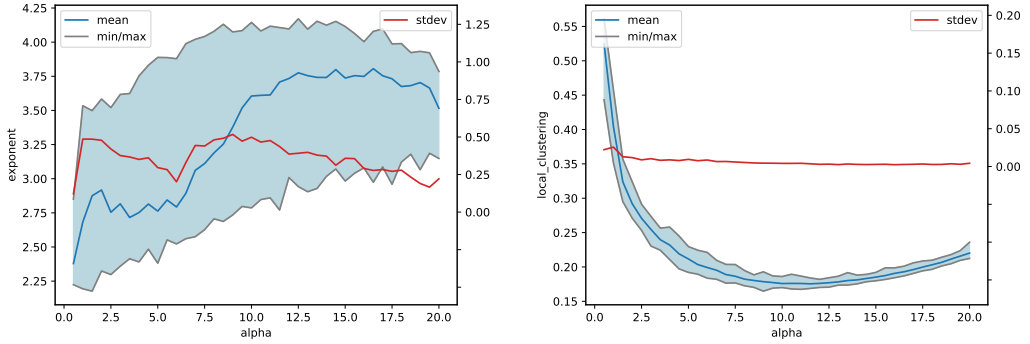


Figure 1: Obtained power-law exponents and average local clustering coefficients for varying edge-price factor α for $n = 10000$ for the global model

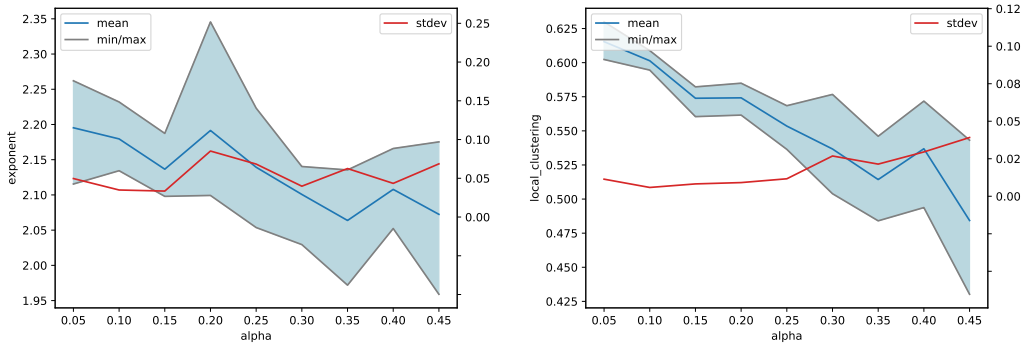


Figure 2: Obtained power-law exponents and average local clustering coefficients for varying edge-price factor α for $n = 20000$ for the local model

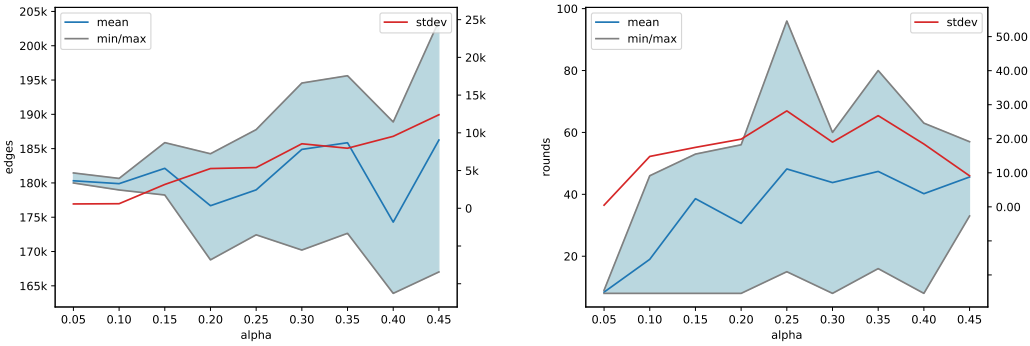


Figure 3: Measured total number of edges and the convergence speed for varying edge-price factor α for $n = 20000$ nodes for the local model

networks if the edge-prices are high. This shows that the majority of edges that are created in earlier steps of the process will be created independently of the edge-price factor and this parameter only influences later rounds where much fewer edges are added to the network. Moreover, also for the global model we observed unexpected behavior when considering the total number of edges. There we get sparser networks for very low and very high alpha.

Another aspect we investigated was the influence of the edge-price factor α on the convergence speed, i.e., the number of rounds needed until the network is built via our incremental process. As expected, we observe a higher number of rounds for high edge-price. The reason is that in the later rounds only very few but crucial edges are built and these late-edges then trigger the addition of other edges later on. Interestingly, also here we get a contrast when looking at the global model. There, the number of rounds seems to be invariant to changing edge-prices.

6 Next Steps

We made significant steps towards a systematic reinvestigation of the relationship between specific parameter settings and obtained network properties. In particular, we established that the desired adjustability of core network properties via the edge-price parameter is still valid but that its influence is weaker than expected. This limits the achievable parameter range for high numbers of network nodes considerably and we will have to adjust the model the cope with this detrimental behavior. We expect that the obtained power-law exponents and the average local clustering coefficients can also be influenced by a modified strategy choice of the agents. Exploring this seems to be a promising next step.

Another main problem of the current models is that the diameter in the obtained networks is rather low. While a diameter of $\Theta(\log n)$ or $\Theta(\log n / \log\log n)$ would be ideal, the experiments suggest that we get a constant diameter instead. Recently we achieved an analytical break-through for one of the models where we could prove that for this model only a constant diameter is possible. This calls for adjusting the model to allow for larger diameter. We already have several ideas of how to achieve this goal and we are eager to put these ideas to the test in large scale experiments.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. "Internet: Diameter of the world-wide web". In: *nature* 401.6749 (1999), page 130.

[2] V. Bala and S. Goyal. "A noncooperative model of network formation". In: *Econometrica* 68.5 (2000), pages 1181–1229.

[3] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[4] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pages 509–512.

[5] K. Bringmann, R. Keusch, and J. Lengler. *Geometric inhomogeneous random graphs*. preprint. 2015. arXiv: 1511.00576 [cs.SI].

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. "Graph structure in the Web". In: *Computer Networks* 33.1 (2000), pages 309–320.

[7] A. Chauhan, P. Lenzner, A. Melnichenko, and L. Molitor. "Selfish Network Creation with Non-Uniform Edge Cost". In: *SAGT'17*. Springer. 2017, pages 160–172.

[8] F. Chung and L. Lu. "The average distances in random graphs with given expected degrees". In: *PNAS* 99.25 (2002), pages 15879–15882.

[9] A. Cord-Landwehr and P. Lenzner. "Network Creation Games: Think Global - Act Local". In: *MFCS'15*. 2015, pages 248–260.

[10] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. "The Price of Anarchy in Network Creation Games". In: *ACM Transactions on Algorithms* 8.2 (2012), page 13.

[11] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. "On a Network Creation Game". In: PODC'03. Boston, Massachusetts: ACM, 2003, pages 347–351.

[12] T. Friedrich and A. Krohmer. "On the diameter of hyperbolic random graphs". In: *ICALP'15*. Springer. 2015, pages 614–625.

[13] J. Kleinberg. "The Small-world Phenomenon: An Algorithmic Perspective". In: *STOC'00*. STOC '00. Portland, Oregon, USA: ACM, 2000, pages 163–170. ISBN: 1-58113-184-4.

[14] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. "Hyperbolic geometry of complex networks". In: *Phys. Rev. E* 82 (3 Sept. 2010), page 036106.

[15] P. Lenzner. "Greedy Selfish Network Creation". In: *WINE'12*. 2012, pages 142–155.

[16] J. Leskovec, J. Kleinberg, and C. Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *SIGKDD'05*. ACM. 2005, pages 177–187.

[17] M. Newman, A.-L. Barabasi, and D. J. Watts. *The structure and dynamics of networks*. Princeton University Press, 2011.

[18] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[19] C. H. Papadimitriou. "Algorithms, games, and the internet". In: *STOC'01*. 2001, pages 749–753.

[20] D. N. Schumann. "Exploring Game Theoretic Models for Generating Real World Networks". Master's thesis. Hasso Plattner Institute, University of Potsdam, 2018.

[21] C. Weyand. "Locality in Game Theoretic Models for Real-World Networks". Master's thesis. Hasso Plattner Institute, University of Potsdam, 2018.

Adversarial Risk Analysis Against Obfuscation Attacks

Alberto Redondo and David Ríos Insua

Instituto de Ciencias Matemáticas, Consejo Superior de Investigaciones Científicas
{alberto.redondo,david.rios}@icmat.es

Malware detection is still an important issue in the current world due to the huge quantity daily attacks. Obfuscation procedures are used by adversaries to create metamorphic malware difficult to detect. These procedures include dead-code insertion, register reassignment, subroutine reordering, code transportation, among others. Adversaries use several attack techniques based on: poison attacks, in which they pretend to mislead the anti-malware detection tool; and, evasion attacks, in which they perturbs the data causing misclassification. In this work, we would determine how obfuscation procedures affect the features extracted from malware and we would build an adversarial model to simulate the adversary behaviour when attackers perform poison and evasion to improve malware obfuscated detection.

1 Introduction

Obfuscation techniques are used by attackers to create malware hard to detect. These techniques have two main forms: metamorphism and polymorphism. Polymorphism methods change the binary code maintaining one of the parts intact in each new copy. Metamorphism methods used to be more advance obfuscation making substitutions, transportations, dead code insertion or register renaming. These methods create a different malware copy making it difficult for anti-malware tools to detect it, disinfect or quarantine because the binary signature will be different in each iteration.

2 Framework

We use a framework based on hybrid analysis which combine static and dynamic features extraction.

The framework extract static features such as operation codes, registers, API calls, entropy, files read, written, deleted, copied, moved, recreated, opened, length of file operations among others and the dynamic features running the binaries in a controlled environment. Once we have extracted the malware features, we use Naïve Bayes (NB) to classify the malware.

2.1 Static features

The features obtained from the binaries are divided into three categories:

ASM We convert the binaries into assembled files through objdump. Then, we extract its sections, registers, Operation Codes (OpCodes), API calls and keywords. We count the number of times that a feature appears in the binary.

Hex dump We transform the binaries into hexadecimal format through hexdump. Suddenly, we obtain the mean, median, maximum, minimum of the binary entropy, along with the entropy of the whole binary, the difference between maximum and minimum and the entropy variance.

PE Header The standard file format allows us to extract features from the binaries such as the size of the code, the number of sections, symbols or imports.

2.2 Dynamic features

Dynamic features are generated based on the run time behaviour of the binaries executed within a Virtual Machine. To perform such analysis we use the [1] tool, which generates a report from the behaviour obtained over a fixed period of time, being 2 minutes the default configuration. We obtain the reports from all the binaries selecting 12 features which we consider that are important for malware detection including the number of mutex, the number of file operations such as the files read or deleted, the register operations and the dll libraries loaded.

2.3 Training

Once we obtain all the features, we may select the algorithms to be used for malware detection. Initially, we consider a Naive Bayes (NB) [2]. We assess the classification results with Detection Accuracy (DA), False Positive (FPR) and False Negative (FNR) Rates.

2.4 Operation

From the above stages, we may setting up the proposed framework into operation, in which we are able to make predictions to detect new malware and updating the model when is required.

3 The problem with obfuscated malware

An attacker could produce obfuscated malware affecting critically the accuracy of the proposed approach for numerous ML algorithms. To illustrate our point, we obfuscate malware through a metamorphic engine called metame. This tool creates new software clones with the same behaviour but different structure: it disassembles the binary seeking for the OpCodes {nop, xor, sub, push, pop, or} and makes some operations with them; for instance, the OpCode xor is replaced by sub. This process modifies only static features but not the dynamic ones as metame aims at preserving

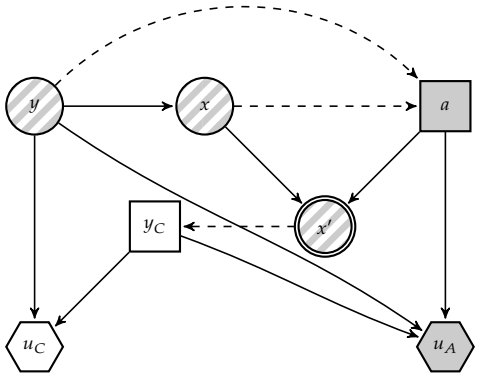


Figure 1: Bi-agent influence diagram for the problem faced by Alan and Cleo

the malware clone behaviour. To observe the impact obfuscation process, we trained the NB classifier with non obfuscated VT (2018) malware mixed with benign binaries (50 % malware, 50 % benign). Then, we predict through a test set formed by obfuscated VT malware mixed with benign binaries. We performed evasion attacks in test time in which we would expect that NB decrease its classification accuracy.

The degradation produced by the obfuscation process decreasing the accuracy around 31 % in 1000 experiments, in which the classifier is almost classifying the binaries randomly.

4 Adversarial Risk Analysis for obfuscation Attacks (AROA)

The model proposed to detect obfuscation attacks adapts the Adversarial classification: An adversarial risk analysis approach, [3]. We sketch the common elements with that approach detailing the differences. We consider a classifier (C, she="Cleo") aiming at maximising her expected utility when classifying benign binaries (y=B) and malware ones (y=M), and an adversary (A, he="Alan") that is willing to obfuscate the binaries to reach benefits maximising his expected utility. Thus, he modifies the features x in binaries to $x' = a(x)$. We designate the transformation x to x' through $a_{x \to x'}$. The problem faced by Alan and Cleo is represented in figure 1 through a Bi-agent influence diagram.

Grey nodes represent issues that affect solely Alan's decisions; white ones those that impact solely Cleo's decision; and, finally, striped nodes affect both agents. Alan's decision is represented through node a (the obfuscation attack chosen) and Cleo's decision through y_C (the classification choice). The impact of obfuscation over x results in x'. Alan and Cleo's utilities are shown through nodes u_A and u_C, respectively. Cleo needs to determine the class y, when observing x' with her guess y_C providing her an utility $u_C(y_C, y)$; Alan, depending on her guess y_C, would obtain his utility $u_A(y_C, y, a)$.

4.1 Cleo's approach

Cleo deals with the problem as a game from a Bayesian perspective. She builds the decision problem taking into account that Alan's decision is random to her. Cleo's required elements are:

- $p_C(y)$, assessing Cleo's conviction about the binary y label with $p_C(M) + p_C(B) = 1$ and $p_C(M), p_C(B) \geq 0$.
- $p_C(x|y)$, models her beliefs about the features of the binary given its class y.
- $p_C(x'|a, x)$, describes Cleo's beliefs about the features transformation through an obfuscation attack over a binary with features x.
- $u_C(y_C, y)$, models Cleo's utilities when she predicts class y_C and the class is y.
- $p_C(a|x, y)$, represents Cleo's beliefs over Alan's action a given a binary with features x and label y.

4.2 Alan's approach

Alan only obfuscates malware binaries, modifying its features x into x' to maximise his expected utility based on making Cleo classify malware binaries as benign. Alan's decision making model requires:

- $p_A(x'|a, x)$, which assess his beliefs about the transformation of the binary when obfuscated.
- $u_A(y_C, y, a)$ describes Alan's utility when Cleo predicts the label to be y_C, the actual label is y and the attack is a.
- $p_A(c(x')|x'))$, assesses Alan's thoughts about Cleo's prediction when she observes the features x' of the obfuscated binary. Let us designate by $p = p_A(c(a(x)) = M|a(x))$ the probability that Alan concedes to Cleo saying that the binary is malware, given that she observes x'. Since Alan will have uncertainty about it, we denote its density by $f_A(p|a(x))$ with expectation $p^A_{a(x)}$.

5 Results

We made large scale of experiments using 1000 features extracted from binaries performing poison attacks and evasion attacks to check this approach. The attacks performed simulates the attacks of a metamorphic engine metame. The results obtained detecting poison and evasion attacks are shown in figure 2. In case of poison attacks, we note that AROA obtains slightly 9 % higher accuracy than NB. In the case of evasion attacks, AROA acquired 12 % higher accuracy than NB. We observe that NB shows some fluctuations altering the accuracy with unbalanced results. AROA presents a stable behaviour contributing with the same accuracy in those cases.

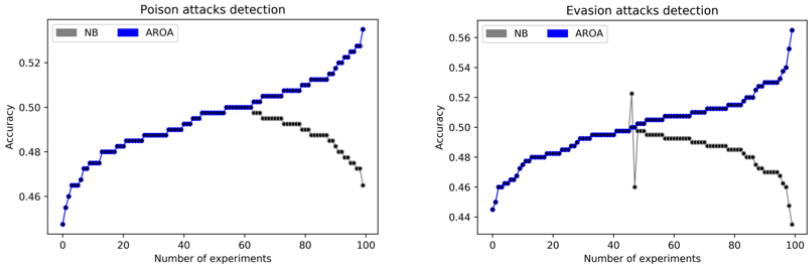


Figure 2: Accuracy NB vs AROA through poison and evasion attacks

6 Conclusions

We explore the performance between NB and AROA model detecting obfuscated malware when the adversary perform poison and evasion attacks. AROA obtains better results than NB getting 12 % higher accuracy in the best case. In addition, AROA presents stable behavior during the detection process in which NB shows variations in several cases. We think this approach could contribute to progress in the field to detect malware obfuscated.

References

[1] Cuckoo. *Cuckoo Sandbox*. 2018. URL: https://cuckoosandbox.org/ (last accessed 2020-04-01).

[2] D. D. Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *Machine Learning: ECML-98*. Edited by C. Nédellec and C. Rouveirol. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pages 4–15. ISBN: 978-3-540-69781-7.

[3] R. Naveiro, A. Redondo, D. R. Insua, and F. Ruggeri. "Adversarial classification: An adversarial risk analysis approach". In: *International Journal of Approximate Reasoning* (2019), pages 133–148.

Behavior-based authentication

Feature engineering and performance evaluation based on large user profiles

Vera Weidmann[1] and Leon Lowitzki[2]

neXenio GmbH
vera.weidmann@nexenio.com, leon.lowitzki@nexenio.com

1 Introduction

Our project contributes to the growing interest of mobile and behavior-based authentication systems. Next to fingerprints and facial features, another meaningful authentication method is a person's gait pattern. Our smartphones are equipped with a variety of sensors and these can measure the environment and behavior surrounding our phones. We carry these phones with us around the clock, even closely attached to our bodies. Accordingly, data corresponding to our body movement is generated. Research shows that these recorded data signals can be shaped to a unique signature of the phone's owner. The authentication accuracy, though, is highly dependent on the users' physiological and environmental circumstances. The more situations that are covered by the user's training state, the more the system's precision is challenged. In this context, statistical investigations and machine learning algorithms are applied.

Our research team at neXenio confronts behavior-, in particular gait-based authentication, with an enormous data set, covering six thousand walking sequences that have been recorded over the last years.

This semester, our second period at Future SOC Lab, we focused on signal processing and feature evaluation.

2 Behavior-based authentication

Nowadays, smartphones can be unlocked via password, fingerprint, or, as more recently introduced, face recognition. The last two methods are biometric authentication methods, which use user related characteristics, e.g. the friction ridges of the finger or facial features, to recognize people.

Nowadays, another biometric generates interest and enthusiasm: gait authentication. Here, the user's motions, such as the leg's forward, backward or sideways movements, are observed by sensors. The next sections introduce the general methodology for gait-authentication as well as neXenio's course of action in this field.

2.1 Methodology for gait-based authentication

Nowadays, gait-based authentication methods are well researched. Especially walking sequences, recorded by a smartphone or smart watches' inbuilt sensors, such as the accelerometer and gyroscope sensor, are reflected. Sprager and Juric [10] as well as Connor [2] summarized the state of the art methodologies used in this field. Guidelines for general data processing are stated by [1] and [4].

In brief, a walking sequence is cut in pieces and summarized by an average gait cycle. This cycle is used as a template and the dissimilarity of a new step is concluded by different distance metrics, such as Euclidean, Hamming, Manhattan or Tanimoto distance. Other approaches use statistical measurements in the time and frequency domain to gather descriptive information about walking sequences. Calculations such as mean, median, standard deviation, third and fourth order cumulants, skewness and kurtosis, and distribution parameters, such as a ten-bin histogram are used ([6], [8], [9]). Likewise, we found features such as the root mean squared, median absolute deviation, average absolute variation, quantiles, interquartile range, correlations and zero-crossing in several papers. In the frequency domain, Fourier and cepstral coefficients, discrete cousin transformations and wavelets are used [10]. Next to principal component analysis, Gait Dynamic Images [13] or geometric template matching were applied to gait sequences [5].

Besides cycle-based assignments, different machine learning techniques are used for authentication: linear and logistic regressions, k-nearest neighbors, support vector machines (SVM), hidden markov models and convolutional neuronal networks.

2.2 seamless.me @ neXenio

neXenio GmbH is a small company located in the middle of Berlin. They take up the challenge of developing high secure IT-solutions that address data sharing and virtual collaboration needs of digital workspaces. One of their products called "seamless.me" targets the idea of behaviour-based authentication. The most important difference to other authentication systems and research approaches is that neXenio's implementation focuses on the premise of data privacy and security. Since data is processed locally on the device itself, data is never sent to external computation resources. Thus, the underlying classification approach is required to be a one-classification problem. An algorithm is deployed that only considers data from a single person. This training set is neither polluted by any outlier nor is enriched by data from other users. The algorithm must detect whether a new unknown walking observation fits to the learned pattern. In general, this type of classification shows less precise results than binary or multi-class classification as the difference between users are not known. Only very few research studies considered this classification type, e.g. by using one-class SVMs.

Another challenge regarding local processing is the device's battery drain. Therefore, neXenio implemented a more efficient outlier technique instead of widely utlilized battery-expensive algorithms. This outlier recognition algorithm is based on multi-level hierarchical nested histograms. Each histogram creates a discrete

frequency distribution of the sensor data's processed features to a specific grain. Features, mentioned in the previous section 2.1, were evaluated and assembled in a way that the best authentication performances were attained.

3 The Problem of learning from a variety of gait patterns

Real-world gait-based authentication applications have to deal with the variety of natural gait forms. If a specific, but genuine form is unknown by the algorithm, it can hardly be assigned to the user. Gait-affecting factors are of physiological or environmental nature. While physiological factors induce more unconscious circumstances, such as permanent gait abnormalities or temporal changes caused by mood, environmental factors are represented by clothing, shoes, surfaces, slopes or obstacles [10]. Real world behavior-based authentication systems need to be robust for long-term utilization as these factors can change by varying degrees from time to time.

Public datasets for gait recognition do not imply a comprehensive overview about a user's possible walking situations. While OU-ISIR Biometric Database of Osaca University [7] is the largest database in the field of gait recognition, holding nearly 750 subjects, just one environmental factor, inclined terrain, was recorded in one session. Frank et al. [5] published another dataset in cooperation with the McGill University. This database contains just 20 participants, but data is recorded in the wild in two distinct sessions, meaning that people could have worn different clothes and shoes per session. Research that relies on this database, detected the effect of clothes the most. In all analyses the authentication performance suffers in cross-day comparison, particularly when people had a major change in trouser type ([5], [11]). This effect is also shown by the larger dataset of Subramanian et al. [12].

Purpose of Future SOC Lab utilization

In the last years, we collected a great amount of data files, covering this variety of physiological and environmental forms. At the moment, our statistical investigations as well as our novelty detector, which we use for user classification, can be just tested by dividing our dataset into smaller subsets. An evaluation that comprises all walking circumstances is not feasible as the limits of our machines according to RAM are reached.

By applying for utilization of the HPI Future SOC Lab we aimed to improve our authentication approach, refine signal preprocessing, feature engineering and modeling in regard to meet stability for wide user profiles.

- Calculation of performance metrics of the current state algorithm by using the total data of six thousand walking samples.
- Parameter tuning for signal preprocessing to find, e.g., the most efficient and battery friendly step-cycle detection algorithm per sensor.

- Evaluation of best feature combinations and model parameters; supported by clustering algorithms and comparison of user's distribution (based on histograms).

4 Future SOC Lab resources

Again, Future SOC Lab provided us with an isolated server, allowing us to process all of our data in parallel. This was one of the main advantages, as combining all our recordings into one pandas data frame consumed over 600 GB RAM — this was not even suitable for processing on the provided server.

So, each signal sequence was tuned in our new pipeline, containing activity recognition (Job 1), filtering and normalization, cycle detection (Job 4) and feature extraction (Job 2, 3). Table 1 summarizes RAM, the number of jobs that run in parallel and time consumption for each task.

Table 1: Future SOC Lab Resources

Job#	Description	RAM	Cores	Duration
1	Data cleaning and walking recognition	16 GB	30	4 h 40 min
2	Feature comparison by histogram similarity	-	20	30 min
3	Cluster analysis and feature evaluation	-	20	36 h
4	Cycle detection and parameter tuning	19 GB	30	23 h 30 min

5 Evaluation

5.1 Performance metrics

For classification objectives, generally, the true positives, false positives, false negatives and true negatives are calculated for evaluating the performance of an algorithm. Based on these four performance classes, certain metrics can be calculated: accuracy, recall, precision, specificity and the F1-score. The latter metric is a combination of recall, how often a genuine walking sequence has been classified correctly over all real positives, and precision, the ratio of correct predicted positive observations to all the predictions.

Additionally, the equal error rate (EER) is included in model selection. This metric reflects the closest point of false matches and false non-matches. While false matches are equivalent to false negatives, where imposters got authenticated, false

non-matches include all false positives plus sequences where nobody could get authenticated.

In addition, neXenio uses another measurement, the fitness score, which is based on a fitness function to measure goal achievement. If all genuine walking sequences are classified correctly and imposters are discarded the fitness score will return 1.

5.2 Findings

A major refactoring of our data pipeline required running all previous conducted analysis and evaluations of the last SOC Lab period (winter term 2018/19) again. We can confirm the old result of an EER of 4 % and a fitness score of 71 %. Compared to other research studies this rate is satisfying.

However, similar to last semester's outcome, we realized that authentication accuracy varies a lot between users. We conducted a cluster analysis that yielded the feature's similarity among different users and gait variations. Additionally, features were ranked by their cluster importance.

Furthermore, we gained a good understanding about a consistent step cycle detection for a person's walking behavior. Five different detection algorithms were tested. Best signal pre-transformations and cycle detection parameters were tuned. Feature engineering and classification outcome are still in progress.

5.3 New challenges

Although we found a robust cycle detection, covering various types of walking setups, one major factor influences the performance the most: the phone's orientation in the trousers pocket. This is a well known problem in research and is addressed by a conversion of the phone's system "... into a resilient and unbiased device- and environment-independent coordinate system" [3]. Furthermore, we plan to assess classifiers for other phone locations like jackets and bags, while also generalizing data for different walking speeds.

6 Conclusion and future work

The resources of HPI's Future Soc Lab enabled us to intensify our data analysis and improve the authentication model. Only by using the advantage of the server's parallel computing, could we process our data — and even speed up the total run time. The main outcome of this semester's term was finding the step-cycle detection method that fits the variety of our walking data.

In the future we aim to refine our authentication approach with more feature engineering, in particular, engineering that solves the problem of the phone's orientation. In this regard, we look forward to using the resources of Future SOC Lab in the next semester again. We want to thank the Future SOC Lab team for the great support.

References

[1] A. Buriro, Z. Akhtar, B. Crispo, and S. Gupta. "Mobile Biometrics: Towards A Comprehensive Evaluation Methodology". In: *2017 International Carnahan Conference on Security Technology (ICCST)*. August. 2017. ISBN: 978-1-5386-1585-0. DOI: 10.1109/CCST.2017.8167859.

[2] P. Connor and A. Ross. "Biometric recognition by gait: A survey of modalities and features". In: *Computer Vision and Image Understanding* 167 (2018), pages 1–27. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2018.01.007.

[3] A. Ferreira, G. Santos, A. Rocha, and S. Goldenstein. "User-Centric Coordinates for Applications Leveraging 3-Axis Accelerometer Data". In: *IEEE Sensors Journal* 17.16 (2017), pages 5231–5243. ISSN: 1530-437X. DOI: 10.1109/JSEN.2017.2723840.

[4] R. Ferrero, F. Gandino, B. Montrucchio, M. Rebaudengo, A. Velasco, and I. Benkhelifa. "On gait recognition with smartphone accelerometer". In: *Proceedings - 2015 4th Mediterranean Conference on Embedded Computing, MECO 2015 - Including ECyPS 2015, BioEMIS 2015, BioICT 2015, MECO-Student Challenge 2015* (2015), pages 368–373. DOI: 10.1109/MECO.2015.7181946.

[5] J. Frank, S. Mannor, J. Pineau, and D. Precup. "Time Series Analysis Using Geometric Template Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3 (Mar. 2013), pages 740–754. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.121.

[6] T. Hoang, D. Choi, and T. Nguyen. "Gait authentication on mobile phone using biometric cryptosystem and fuzzy commitment scheme". In: *International Journal of Information Security* 14.6 (2015), pages 549–560. ISSN: 1615-5270. DOI: 10.1007/s10207-015-0273-1.

[7] T. T. Ngo, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi. "The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication". In: *Pattern Recognition* 47.1 (2014), pages 228–237. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2013.06.028.

[8] C. Nickel. "Accelerometer-based Biometric Gait Recognition for Authentication on Smartphones". PhD thesis. Darmstadt: Technischen Universität Darmstadt, June 2012. URL: http://tuprints.ulb.tu-darmstadt.de/3014/.

[9] C. Nickel and C. Busch. "Classifying accelerometer data via hidden Markov models to authenticate people by the way they walk". In: *IEEE Aerospace and Electronic Systems Magazine* 28.10 (2013), pages 29–35. ISSN: 0885-8985. DOI: 10.1109/MAES.2013.6642829.

[10] S. Sprager and M. Juric. "Inertial Sensor-Based Gait Recognition: A Review". In: *Sensors* 15.12 (Sept. 2015), pages 22089–22127. ISSN: 1424-8220. DOI: 10.3390/s150922089.

[11] S. Sprager and M. B. Juric. "An Efficient HOS-Based Gait Authentication of Accelerometer Data". In: *IEEE Transactions on Information Forensics and Security* 10.7 (2015), pages 1486–1498. ISSN: 1556-6013. DOI: 10.1109/TIFS.2015.2415753.

[12] R. Subramanian, S. Sarkar, M. Labrador, K. Contino, C. Eggert, O. Javed, J. Zhu, and H. Cheng. "Orientation invariant gait matching algorithm based on the Kabsch alignment". In: *2015 IEEE International Conference on Identity, Security and Behavior Analysis, ISBA 2015* (2015), pages 1–8. DOI: 10.1109/ISBA.2015.7126347.

[13] Y. Zhong, Y. Deng, and G. Meltzner. "Pace independent mobile gait biometrics". In: *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems, BTAS 2015* (2015), pages 1–8. DOI: 10.1109/BTAS.2015.7358784.

Human Motion Analysis in Daily Life
A Low-Cost and Unobtrusive Gait Analysis System

Justin Albert, Lin Zhou, and Bert Arnrich

Connected Healthcare
Digital Health Center, Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

Gait is an important indicator of a person's health status. Measuring gait usually needs to take place in a dedicated gait laboratory, with expert knowledge and complex subject preparation. For many years, the development of alternative gait analysis systems has been an active research topic. The aim of this work is to present a low-cost, mobile and unobtrusive gait analysis method that is able to work without any preparation of the subject. Evaluation is done by comparing the results with those reported in the existing literature.

1 Introduction

Human motion analysis is an important instrument for the assessment of neurological diseases such as Parkinson's disease (PD) or stroke. Motor and cognitive abilities can be derived from a patient's gait. Usually, patients are examined for early detection or progress monitoring of neurological diseases in a special gait analysis laboratory with expensive equipment such as the Vicon system[1] or Optogait.[2] The Vicon multi-camera system, which is regarded as the so-called gold standard system, requires active or passive markers to be attached onto the patient. Therefore, these gait tests must be performed in gait analysis laboratories and are not suitable for in-the-wild assessments.

In physiotherapy, e.g. stroke rehabilitation, patients are monitored and treated by a physiotherapist, but after discharge from the rehabilitation centre they are often isolated at home without supervision. Without any monitoring, it is difficult to continuously quantify the individual's post-release progress, e.g. to predict an increased risk of falling. This problem raises the need for novel, unobtrusive measurement methods that can work at home without expert knowledge and operators. Common measurement systems like Inertial Measurement Units (IMUs) or inexpensive camera solutions like Microsoft Kinect or smartphones offer the possibility to measure human movements without having to prepare the patient or purchase expensive special equipment. Due to recent successes in machine learning, especially in deep learning, the topic of markerless human motion analysis from two-dimensional

[1] https://www.vicon.com (last accessed 2020-04-01).
[2] https://www.optogait.com (last accessed 2020-04-01).

images using convolutional neural networks (CNNs) has been investigated. By applying this novel technology in the medical field, it would provide a cost-effective and unobtrusive gait analysis that can be performed outside the laboratory.

2 Related Work

Automated gait analysis has been a research focus of computer science for many years. The goal to develop a cost-effective gait analysis system has produced a multitude of gait analysis systems, e.g. by utilizing force plates, rgb or depth cameras. A well researched device for gait analysis is the Microsoft Kinect camera, which was developed as a gaming controller for the Microsoft Xbox console [11]. Many studies have shown that the Kinect camera provides reasonable results for a subset of spatiotemporal gait parameters [9]. In one study by [8], the Microsoft Kinect camera was used to identify and segment gait cycles in order to predict specific gait parameters. Another study was utilizing the Kinect to evaluate the gait of a patient walking on a treadmill [1].

Due to the large, publicly accessible data sets for numerous image processing tasks, great progress has also been made in the field of 2D and 3D human pose estimation. The state-of-the-art multi-person 2D human pose estimation method from RGB monocular images is the so-called OpenPose library [2], which surpassed all other network models in the Coco 2016 keypoints challenge [4]. The network consists of a two-step procedure that first predicts all relevant keypoints and then a map that contains information about the relationship of adjacent joint connections (Part Affinity Fields). Another research group [5] developed a network for 3D position estimation, which first learns 2D positions and then elevates these 2D locations into 3D space. Another 3D human pose estimation method proposed by [6] first estimates the 2D points and 3D points and then uses a postprocessing step to fit a defined skeleton model into the predicted keypoints.

3 Method

The system developed here consists of a two-stage process. The first phase is a 2D skeleton tracking process with machine learning to locate human joints in monocular images. The second phase is an analysis step that processes the extracted joint positions in order to obtain gait variability.

3.1 2D Human Pose Estimation

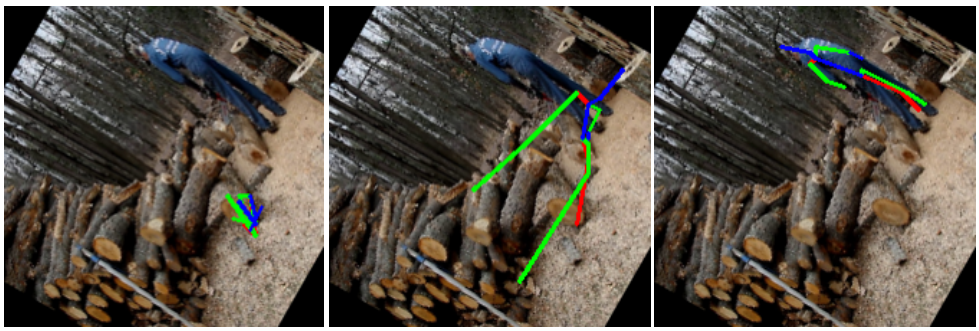
The 2D human pose estimation method developed here is based on the common technique of predicting 2D heatmaps, which indicate the joint positions to the corresponding locations from the input RGB image. The system consists of a CNN that takes a certain image $I \in \mathbb{R}^{w \times h}$ with width w and height h and predicts a

heatmap $H \in \mathbb{R}^{w \times h \times j}$, where $j = 0, \ldots, J$ refers to an individual joint. The ground truth heatmaps $\hat{H}_j$ are generated by using a 2D Gaussian at the center of the corresponding (x, y) joint position. By using this approach larger values in the heatmap indicate a higher confidence of the network that the joint position is at the corresponding location in the input image. Previous work has shown that predicting heatmaps has a better performance than regressing the (x, y) joint locations directly. The objective function to learn the heatmaps is the Mean-Squared-Error (MSE) function, which calculates the error between the predicted heatmaps H_j and the ground truth heatmaps $\hat{H}_j$, as shown in equation (1).

$$MSE = \frac{1}{J} \sum_{j=0}^{J} \left\| \hat{H}_j - H_j \right\| \tag{1}$$

The model developed here has the popular ResNet50 architecture [3] as its backbone, which has proven itself in the task of image classification. The ResNet50 model was adapted by removing the last fully-connected layers and adding transposed convolutional layers instead to produce heatmaps with the same spatial dimensions as the input images. The final joint locations $L_j \in \mathbb{R}^2$ are determined by finding the locations in the heatmaps that have the highest confidence values.

In order to increase the robustness of the skeleton tracking, data augmentation techniques during training were used. For all training batches a random scaling of a factor $s \in [1, 1.3]$ and a random rotation of $\alpha \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ radians was applied. The empty corners of the rotated images were left black. As a pre-processing step, all images were scaled to the interval $[-1, 1]$ to avoid exploding or vanishing gradients. Figure 1 shows the training process of the network in the t^{th} training iteration by visualizing the locations with highest confidence and connecting the joints using the skeletal topology. The figure shows that the network predictions constantly come closer to the optimal solution.



(**a**) $t = 1$ (**b**) $t = 2600$ (**c**) $t = 10000$

Figure 1: Visualization of the training progress at the t^{th} training iteration. The image is rotated due to the data augmentation strategy.

3.2 Estimating Temporal Gait Parameters

Based on the 2D keypoint detection model from the previous section, the goal is to estimate gait parameters that quantify a person's walking behavior. Therefore, the keypoints of a person are tracked over time by analyzing a video, e.g. from a smartphone camera, to obtain a temporal profile of walking patients. Since the 2D keypoints are present in pixel coordinates, the gait parameters derived here are only temporal characteristics. This means that the gait parameters considered here are the stance and swing phase duration as well as the duration of the entire gait cycle. These parameters can be further processed, e.g. to obtain the variability of gait cycles between the left and right foot.

The setup of the system consists of a treadmill as well as a stationary mounted camera, which films the subject from a 90° angle. For the calculation of the gait cycle duration the image is regarded as a 2D Cartesian coordinate system, where the origin is the upper left corner and the y-axis expands in height and the x-axis expands in width. The user is standing on the treadmill and walking in positive x direction (to the right). In this scenario, the left and right ankle keypoints are tracked while walking on the treadmill. When plotting the x components from the ankles over time (refer to figure 2), a sinusoidal pattern results due to the constantly moving treadmill belt [10]. The peaks of the curve indicate the point where the heel touches the ground, while the valleys indicate the initialization of the swing phases. By calculating the velocity vectors $V = \{\vec{v}_1, \dots, \vec{v}_N\}$ of the curve with N data points, a sign change of the x-component indicates a change in the swing or stance phase. The velocity vectors $\vec{v}_t$ are calculated as difference of consecutive ankle locations $X = \{x_1, \dots, x_N\}$ as shown in equation (2).

$$\vec{v}_{t+1} = x_{t+1} - x_t \tag{2}$$

A sign change in the x component of the velocity vector indicates whether the user is swinging the leg in the positive x-direction or the foot touches the ground and is pulled back by the running treadmill belt. For easier data processing the time series of velocity vectors is transformed into a binary signal $B = \{b_1, \dots, b_N\}$ by applying the function as shown in equation (3).

$$b_t = \begin{cases} 1, & \text{if } \vec{v}_t(x) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The swing and stance phase duration can now be estimated by counting consecutive zeros or ones respectively, and dividing this number by the video frame rate.

4 Evaluation

In this chapter, the estimated temporal gait parameters by the system are compared versus common gait properties reported by the literature. As mentioned in [7], the gait cycle in healthy subjects consists of 60 % stance and 40 % swing phase. The

evaluation was done by recording data of a single subject walking on a treadmill with a constant speed of 3 km/h while being filmed with an Apple iPhone 7+ at 24fps. The left and right ankle joints were tracked using the OpenPose [2] library, which currently delivers more accurate results as the here developed method.

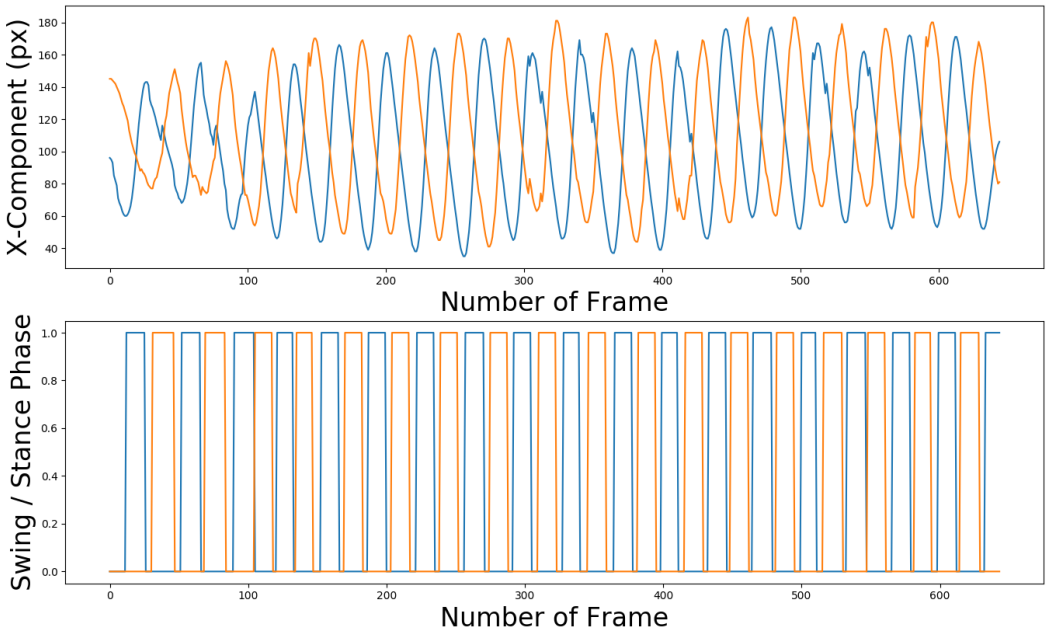


Figure 2: Gait data from one healthy subject. Upper image: Resulting sinusoidal pattern when tracking both ankles. Lower image: Binary signals indicating stance and swing phases.

Figure 2 shows the x components of left and right ankles tracked over approx. 650 frames (approx. 27 seconds). A sinusoidal and symmetric gait pattern can be noticed. Based on this data, the velocity vectors and the subsequent binary signal were calculated. The original data stream was filtered using a mean filter with a window size of $w = 9$. The resulting binary signal was further median filtered with a window size of $w = 9$. Based on the binary signal, the consecutive gait phases were calculated which lead to an average stance time of $\bar{x} = 0.84s, \sigma^2 = 0.006$ as well as an average swing time of $\bar{x} = 0.5s, \sigma^2 = 0.001$ for both feet. This result aligns with those parameters reported in the literature, as the ratio of stance and swing phase is equal to 60 % (0,604).

5 Conclusion

The system presented in this paper can be used for unobtrusive measurements of gait variability, e.g. as an early indicator of neurological diseases. However, the system is in a very early development stage and needs major improvement as well as a more advanced evaluation. The possibly largest limitation of the system is the pixel coordinate system which only can be used for estimating temporal gait parameters instead of accurate physical measurements. A second limitation is that the skeleton tracking so far only works in the two-dimensional space. Possible solutions are either utilizing the dual camera or time-of-flight cameras in novel smartphone models or developing advanced deep learning models capable of accurate 3D joint predictions.

6 Future Work

This work has shown that in principle it is possible to utilize only a smartphone camera for taking optical measurements to quantify the gait variability of a patient. In the future the system should be improved by also including spatiotemporal gait parameters in the feature set, as well as predicting 3D coordinates. Furthermore, the system could provide real-time feedback of gait quality which requires online processing on the device. Also, the accuracy of the system should be compared against a gold standard device such as the Vicon camera system.

References

[1] E. Auvinet, F. Multon, C.-E. Aubin, J. Meunier, and M. Raison. "Detection of gait cycles in treadmill walking using a Kinect". In: *Gait and Posture* 41.2 (2015), pages 722–725. ISSN: 1879-2219. DOI: 10.1016/j.gaitpost.2014.08.006.

[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pages 1302–1310.

[3] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2016), pages 770–778. ISSN: 1063-6919. DOI: 10.1109/CVPR.2016.90.

[4] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. "Microsoft COCO: Common Objects in Context". In: *CoRR* (2014). arXiv: 1405.0312 [cs.CV].

[5] J. Martinez, R. Hossain, J. Romero, and J. J. Little. "A Simple Yet Effective Baseline for 3d Human Pose Estimation". In: *Proceedings of the IEEE International Conference on Computer Vision* (2017), pages 2659–2668. ISSN: 1550-5499. DOI: 10.1109/ICCV.2017.288.

[6] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M.-H. Shafiei, H.-p. Seidel, W. Xu, D. Casas, and C. Theobalt. "VNect". In: *ACM Transactions on Graphics* 36.4 (2017), pages 1–14. ISSN: 0730-0301. DOI: 10.1145/3072959.3073596. arXiv: 1705.01583 [cs.CV].

[7] J. Perry. "Gait Analysis: Normal and Pathological Function". In: *Journal of Sports Science & Medicine* 1.2 (2010), pages 1–551.

[8] A. P. Rocha, H. Choupina, M. Vilas-Boas, J. M. Fernandes, and J. Cunha. "System for automatic gait analysis based on a single RGB-D camera". In: *PLoS ONE* 13.8 (2018), pages 1–24. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0201728.

[9] S. Springer and G. Y. Seligmann. "Validity of the kinect for gait assessment: A focused review". In: *Sensors* 16.2 (2016), pages 1–13. ISSN: 1424-8220. DOI: 10.3390/s16020194.

[10] J. A. Zeni, J. G. Richards, and J. S. Higginson. "Two simple methods for determining gait events during treadmill and overground walking using kinematic data". In: *Gait and Posture* 27.4 (2008), pages 710–714. ISSN: 0966-6362. DOI: 10.1016/j.gaitpost.2007.07.007.

[11] Z. Zhang. "Microsoft Kinect Sensor and Its Effect". In: *IEEE MultiMedia* 19.2 (Apr. 2012), pages 4–10. ISSN: 1070-986X. DOI: 10.1109/MMUL.2012.24.

Measurement-Based Software Performance Engineering for Microservices and Multi-Core Systems

André van Hoorn, Markus Frank, and Henning Schulz

Institute of Software Technology
University of Stuttgart

This report provides a summary of our project "Measurement-Based Software Performance Engineering for Microservices and Multi-Core Systems" conducted during the HPI Future SOC Lab period spring 2019, as well as ideas for a follow-up project for the upcoming period.

1 Project Idea

Our project was divided into two subprojects, namely (1) "DevOps-oriented Load Testing for Microservices" and (2) "Software Performance Engineering for Multi-Core Systems". Both subprojects are a direct continuation of the works that we started in the previous periods. For both subprojects, in order to conduct large-scale experimental evaluations, we need a state-of-the-art computing infrastructure such as the one provided by the HPI Future SOC Lab.

In the remainder of this report, we will provide some more details about the project context (section 1.1 and 1.2) list the granted Future SOC Lab resources (section 2), provide a brief description of our findings (section 3), and outline next steps (section 4).

1.1 DevOps-oriented Load Testing for Microservices

Modern software engineering paradigms and technologies — such as DevOps [2] (including automation as part of continuous delivery) and microservices [13] — are gaining more and more attraction in the software and services engineering communities. Of particular interest are quality-of-service concerns, for instance, w. r. t. performance and reliability. While established approaches for classic contexts (i.e., which do not use DevOps and microservices) exist, their adoption to DevOps and microservices requires considerable research efforts [3, 11].

In the recent years, our group has already contributed architecture-aware approaches for performance and reliability, involving a combination of measurement-based and model-based techniques [12, 14, 17]. In this subproject, we investigate how these techniques can be used in DevOps and microservice contexts — with a particular focus on load testing as a key performance engineering activity [4, 15, 17].

In the past periods, we have already conducted experiments in the HPI infrastructure (e.g., [1]). The activities on load testing conducted during this period were a direct continuation of the activities started during the previous periods. Particu-

larly, our plan was to continue with our experiments on modularization of load tests, which we started in the previous period and sketched in the previous report.

1.2 Software Performance Engineering for Multi-Core Systems

Multicore systems are a permanent part of our daily life. Regardless of whether we consider nowaday's desktop PCs, notebooks, or smart phones—all devices are running on multicore CPUs. To use these hardware features in an efficient way, developers need to build parallel-enabled software. However, the development of such software is more complex than developing sequential software.

To handle the rising complexity, it is necessary to develop software in an engineering-like way. In such a process, software architects plan and analyze software designs on a model level. Software architects can use tools like Palladio to simulate and analyze early-phase software designs. Unfortunately, current approaches and tools lack the ability to consider multicore systems. Therefore, in this project, we aim to find performance prediction methods for multicore systems in the context of our ongoing research [6, 7, 8, 9]

In the previous periods, we have started to use the HPI infrastructure to study performance properties for performance predictions of multi-core systems (e.g., [5]).

The plan for this period was to conduct further experiments with different configurations (e.g., thread numbers and thread pool sizes) to assess what is their impact on performance properties for different use cases or scenarios (e.g., benchmarks) to obtain performance curves. These performance curves will be used by software architects to easily adopt performance prediction models.

2 Used Future SOC Lab resources

We requested and received dedicated (root) access to the following computing resources (servers): *i.*) 896 GB RAM, 80 cores; *ii.*) 32 GB RAM, 24 cores. Dedicated access has been given to us due to our expected high resource demands.

3 Findings

We have worked on the previously stated goals in both subprojects. In this section, we will provide a summary of the experiments and results for both subprojects.

3.1 DevOps-oriented Load Testing for Microservices

We have developed a new approach to automatically generate microservice-tailored load tests for session-based systems from production monitoring data. A publication [16] describing the approach has been accepted for the 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecom-

munication Systems (MASCOTS 2019). The experiments for the paper have been conducted in the HPI infrastructure.

Following our experimental setup described in the previous report, we executed several experiments with the Sock Shop microservices demo application.[1] The goal was to investigate the impact of different tailoring algorithms on the representativeness and the efficiency of the load tests. As a selected result, figure 1 shows the accuracy metric (D) for three different algorithms (request-based, log-based, and model-based) and an untailored workload. Lower values of D indicate a higher representativeness. For our approaches, D is only slightly greater than for the untailored test, but clearly less than for the request-based tests. For details on the experiments, we refer to the publication [16].

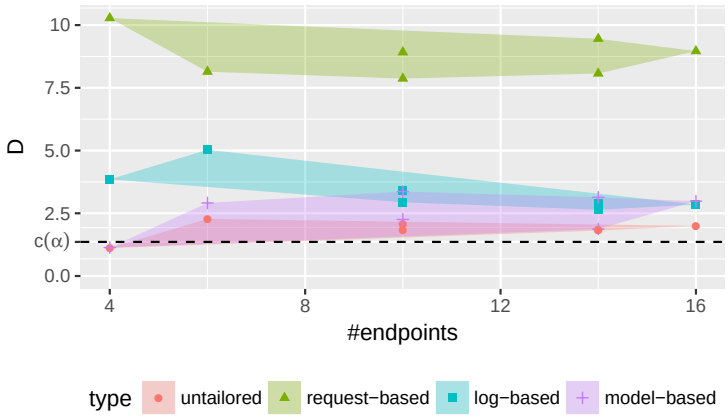


Figure 1: Aggregated representativeness statistic D [16]

3.2 Software Performance Engineering for Multi-Core Systems

We have developed an approach for modeling and predicting memory behavior in parallel systems. A publication [10] describing the study has been accepted for the 10th Symposium on Software Performance (SSP 2019). The experiments for the paper have been conducted in the HPI infrastructure.

We have conducted a series of experiments using the Memtest86[2] benchmark to assess the accuracy of the predictions. Figure 2 shows the accuracy of the predictions in percentage. We distinguish between the two machines (M1 and M2) and different numbers of core, and compare our approach to the simulations without the memory model in place. For M2 we can report an overall increase of accuracy. For M1 we can

[1] https://microservices-demo.github.io/ (last accessed 2020-04-01).
[2] https://www.memtest86.com/ (last accessed 2020-04-01).

only report a great increase for 16 cores. For details on the experiments, we refer to the publication [10].

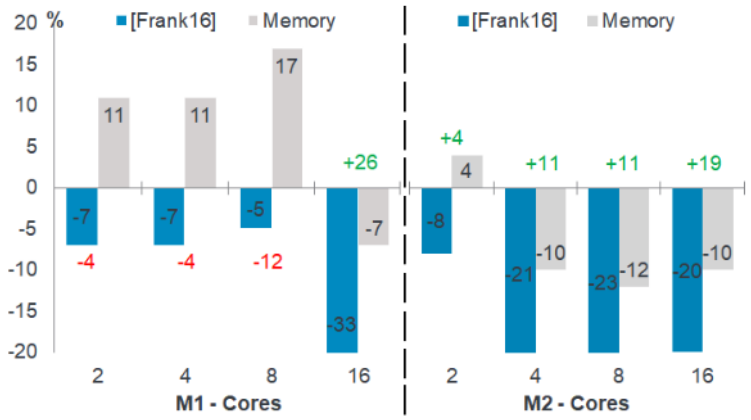


Figure 2: Comparison of prediction accuracy [10]

4 Next steps

We will continue our ongoing research in the two subprojects. To have the ability to execute extensive experiments, we will apply for the next HPI Future SOC Lab period. We plan to extend our load testing approaches to Functions-as-a-Service (FaaS). Extensions for the multi-core experiments include different configurations and more complex parallel programs.

Acknowledgment

We are thankful to the HPI Future SOC Lab for having granted us access to the computing infrastructure. The environment eases the joint work of different organizations on the platform, which has so far been hindered by university-internal access constraints—apart from the fact that an equipment comparable to that of the HPI Future SOC Lab has not been available to us, and that enables extensive performance configuration tests needed to reach our goals.

References

[1] A. Avritzer, V. Ferme, A. Janes, B. Russo, H. Schulz, and A. van Hoorn. "A quantitative approach for the assessment of microservice architecture deployment alternatives using automated performance testing". In: *Proceedings of the 12th European Conference on Software Architecture (ECSA 2018)*. LNCS. Springer, 2018.

[2] L. J. Bass, I. M. Weber, and L. Zhu. *DevOps — A Software Architect's Perspective*. SEI series in software engineering. Addison-Wesley, 2015. ISBN: 978-0-13-404984-7.

[3] A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. Herbst, P. Jamshidi, R. Jung, J. von Kistowski, A. Koziolek, J. Kroß, S. Spinner, C. Vögele, J. Walter, and A. Wert. *Performance-oriented DevOps: A Research Agenda*. Technical report SPEC-RG-2015-01. SPEC Research Group — DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), 2015.

[4] V. Ferme and C. Pautasso. "Integrating Faban with Docker for Performance Benchmarking". In: *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016)*. ACM, 2016.

[5] M. Frank, S. Becker, A. Kaplan, and A. Koziolek. "Performance-influencing Factors for Parallel and Algorithmic Problems in Multicore Environments". In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19)*. To appear.

[6] M. Frank and M. Hilbrich. "Performance Prediction for Multicore Environments — An Experiment Report". In: *Proceedings of the Symposium on Software Performance (SSP 2016)*. 2016.

[7] M. Frank, M. Hilbrich, S. Lehrig, and S. Becker. "Parallelization, Modeling, and Performance Prediction in the Multi-/Many Core Area: A Systematic Literature Review". In: *Proceedings of the 7th IEEE International Symposium on Cloud and Service Computing (SC2 2017)*. 2017.

[8] M. Frank, F. Klinaku, and S. Becker. "Challenges in Multicore Performance Predictions". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE 2018)*. ACM, 2018.

[9] M. Frank, S. Staude, and M. Hilbrich. "Is the PCM Ready for ACTORs and Multicore CPUs? — A Use Case-based Evaluation". In: *Proceedings of the 8th Symposium on Software Performance (SSP 2017)*. 2017.

[10] P. Gruber and M. Frank. "Modelling and Predicting Memory Behavior in Parallel Systems with Network Links: Palladio-based Experiment Report". In: *Proceedings of the 10th Symposium on Software Performance (SSP 2019)*. 2019.

[11] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger. "Performance Engineering for Microservices: Research Challenges and Directions". In: *Companion of the 8th ACM/SPEC International Conference on Performance Engineering* (*ICPE 2017*). ACM, 2017.

[12] A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Kiel Computer Science Series 2014/6. Dissertation, Faculty of Engineering, Kiel University. Kiel, Germany: Department of Computer Science, Kiel University, 2014.

[13] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[14] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske. "Hora: Architecture-aware online failure prediction". In: *Journal of Systems and Software* (2018). DOI: 10.1016/j.jss.2017.02.041.

[15] H. Schulz, T. Angerstein, and A. van Hoorn. "Towards Automating Representative Load Testing in Continuous Software Engineering". In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (*ICPE 2018*) *Companion* (*7th International Workshop on Load Testing and Benchmarking of Software Systems, LTB 2018*). ACM, 2018.

[16] H. Schulz, T. Angerstein, D. Okanović, and A. van Hoorn. "Microservice-tailored Generation of Session-based Workload Models for Representative Load Testing". In: *Proceedings of the 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (*MASCOTS 2019*). 2019.

[17] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, and H. Krcmar. "WESS-BAS: Extraction of Probabilistic Workload Specifications for Load Testing and Performance Prediction—A Model-Driven Approach for Session-Based Application Systems". In: *Journal on Software and System Modeling* (*SoSyM*) (2018).

GPU-Accelerated Causal Structure Learning Beyond the GPU Device Memory Capacity

Christopher Schmidt, Johannes Hügle, and Matthias Uflacker

Enterprise Platform and Integration Concepts
Hasso Plattner Institute for Digital Engineering
{firstname.lastname}@hpi.de

The knowledge on causal structures between variables of a system is relevant for research in various domains. For example in medicine, gene regulatory networks are a practical embodiment of systems biology and can be applied in diagnostics, or drug design. The gene regulatory networks are derived from high-dimensional gene expression data. Processing the data leads to long execution times.

Previous work on GPU-accelerated causal structure learning has shown that the massively parallel processing power of GPUs can significantly reduce the long execution times. Yet, existing GPU-accelerated implementations are restricted to datasets that entirely fit into a single GPU's memory.

In this technical report, we provide detail on a block-wise extension of an existing GPU-accelerated causal structure learning algorithm. We use GPUs provided by the Future SOC Lab to experimentally evaluate our block-wise extension, showing that datasets exceeding the GPU memory can be processed efficiently.

1 Introduction and Background

Learning causal structures from observational data is an active field of research in statistics and data mining. Discovering the causal relationships between observed variables in complex systems supports human understanding and is of particular interest in the context of high-dimensional settings. For example in genetic research, inferring gene regulatory networks from gene expression data supports drug design or diagnostics [8].

Constraint-based algorithms, such as the PC algorithm developed by Spirtes et al. [12] and its order-independent extension, PC-stable algorithm [2], enable the derivation of causal structures from observational data. These algorithms aim to find all possible DAGs, in which nodes refer to the variables and connecting edges represent existing causal relationships [6]. For constraint-based algorithms, the graphical model is derived through the repeated application of CI tests. The appropriate CI test is determined by the data distribution of the involved variables [3]. According to the framework for parallel constraint-based learning [11], the PC algorithm is separated into three phases. In the first phase the Markov blankets are learnt, which is optional and not of interest in this work. In the second phase the neighbors are learnt, and an undirected skeleton graph representing the existing causal relationships is

determined. This phase is referred to as skeleton discovery or adjacency search. In the third phase, the edges within this skeleton graph are oriented to determine the CPDAG. For details, we refer the reader to Sections 2 and 3 of our previous work [10].

In the context of gene expression data, variables are commonly assumed to be multivariate normal distributed yielding to CI tests on the basis of the partial correlations [8]. While a single CI test is computational feasible, the total number of CI tests increases, in the worst case, exponential with regard to the number of variables in the dataset [12]. Under the reasonable assumption that the underlying causal graphical model is sparse, the complexity is reduced to be polynomial [4]. The resulting execution time, in particular for the adjacency search, has been subject to previous research on GPU-acceleration, in which significant speed-up compared to Central Processing Unit (CPU)-based versions is shown [10, 13]. Yet, datasets for analysis in genetic research are often high-dimensional, e.g., see The Cancer Genome Atlas (TCGA) [1]. The corresponding gene expression datasets from TCGA contain information on more than 55 000 genes [7] which exceeds the device memory of recent GPU generations and prevents causal structure learning with current implementations.

With our work, we address this shortcoming of currently existing GPU-accelerated causal structure learning algorithms and provide details on a block-wise extension of the second phase, the adjacency search, that overcomes the device memory limitation. We experimentally evaluate our approach using NVIDIA K80 GPUs, provided by the Future SOC Lab, which are equipped with Tesla GK210 cards having 12 GB of on-chip, device, memory [5].

2 A Block-Wise Adjacency Search on the GPU

The adjacency search of the original PC-stable algorithm determines the undirected skeleton graph C and corresponding separation sets Sepset $\subseteq \mathbf{V} \setminus \{V_i, V_j\}$ for a set of N variables $\mathbf{V} = \{V_1, \dots, V_N\}$. Given multivariate normal distributed data the algorithm processes the correlation matrix Cor of dimension $N \times N$, which contains pre-calculated sample correlation coefficients of all variable combinations. The algorithm uses CI tests to determine the independence of a pair of variables V_i, V_j, $i, j \in 1, \dots, N$ given a separation set Sepset with respect to a significance level α. Note, that the significance level α can be treated as a tuning parameter influencing the sparsity of the estimated skeleton.

Based on the adjacency search of the original PC-stable algorithm, we propose our block-wise version for the GPU, as depicted in algorithm 1. In addition to the input parameters of the original PC-stable implementation for multivariate normal distributed data, it requires a block size *bs*. In a previous report, we proposed to calculate the block size *bs* based on the following constraints. First, the size in memory for each block is required to be less than the memory available on the GPU and is restricted by the number of additional blocks required for CI tests, which need to fit in GPU device memory at the same time. Second, the size of a block should pro-

Algorithm 1: Block-wise adjacency search of PC-stable algorithm on GPU
Input: Vertex set V, correlation matrix Cor, tuning parameter α, block size bs
Output: Estimated skeleton C, separation sets Sepset

1: Start with fully connected skeleton C and $l = -1$
2: **repeat**
3: $l = l + 1$
4: **for all** Vertices V_i in C **do**
5: Let $a(V_i) = adj(C, V_i)$;
6: **end for**
7: $blocks = Split(V, \text{Cor}, C, \text{Sepset}, a, bs)$
8: **for all** b in $blocks$ **do**
9: Transfer b to GPU
10: **if** $l == 0$ **then**
11: $BlockwiseCITest(b, \alpha)$
12: **else**
13: $sepsetblocks = SepSetCombination(b, l, blocks)$
14: **for all** s in $sepsetblocks$ **do**
15: Transfer s to GPU
16: $BlockwiseCITest(b, s, \alpha)$
17: **end for**
18: **end if**
19: Transfer b from GPU
20: **end for**
21: $Merge(blocks)$
22: **until** each adjacent pair V_i, V_j in C satisfy $|a(V_i) \setminus \{V_j\}| \leq l$
23: **return** C, Sepset

vide enough parallel work per block to occupy the GPU Streaming Multiprocessors (SMs).

The block-wise version introduces the following additional steps, compared to the original version of the adjacency search. First, a copy of the current skeleton C is introduced, by calculating the adjacent vertices for each variable V_i, see line 5 in algorithm 1. This allows changes to be handled in parallel without the need for synchronization, until the merge of *blocks*, compare line 21 in algorithm 1. The *blocks* processed within the current level l are computed in $Split()$. The function returns all blocks with the given block size bs. Each block contains an extract of the correlation matrix *Cor*, the list of separation sets Sepset, the skeleton C and the adjacency structure a corresponding to the variables covered within the block. In order to compute the CI tests on the GPU, each block $b \in blocks$ is transferred to the GPU device memory and the covered variables within b are tested for conditional independence. Depending on the current level, additional blocks are required to allow for separation sets. The function $SepSetCombination()$ computes the separation set blocks, which are also transferred to GPU device memory and accessed during

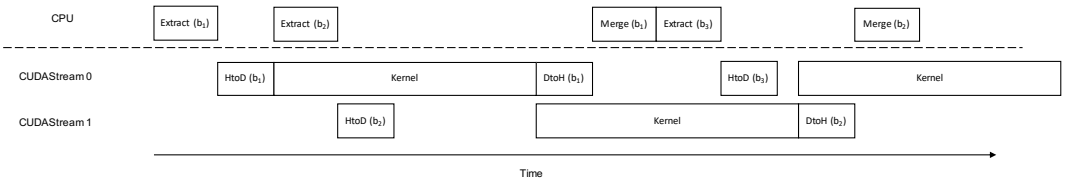


Figure 1: Overlapping execution on CPU with data transfer and execution on GPU within the adjacency search of the block-wise PC algorithm [9]

the CI tests. After the processing of a block b, its data is transferred back to the host memory. Once, all blocks b have been processed, within the current level l, the results are merged, before the next level $l + 1$ is started. This process is repeated, until no more separation set can be formed for the current level l.

Our implementation of the block-wise algorithm is based on previous work [10], which focuses on levels $l = 0, 1$. This enables to investigate the performance of the block-wise algorithm for CI tests with $(l = 1)$ and without $(l = 0)$ a separation set. Further, based on the recent publication of cupc [13] we expect a similar behaviour for levels $l \geq 2$ compared to level $l = 1$. The implementation of algorithm 1 incorporates the following optimizations. First, in level $l = 0$, we do not transfer the adjacency structure $a(V_i)$ and the separation sets Sepset, which reduces the memory requirements. In level $l = 0$, the deletion of an edge is directly carried out on the skeleton C as no synchronization is required. Furthermore, for edges removed in level $l = 0$, an empty separation set is stored, which is the initial value in the data structure Sepset for each pair of variables V_i, V_j. Thus, setting the value is not required during kernel execution for $l = 0$.

Second, we overlap computations on CPU with computations on GPU, as well as, with data transfer between host and device. In particular, the block extraction and merge on the CPU and the data transfer to the GPU are overlapped with kernel execution on GPU, as shown in figure 1. This reduces overall runtime of the block-wise algorithm. Implementing the optimization, we slightly adapt the proposed algorithm 1. Instead of extracting all *blocks* within the function $Split()$, the function returns a mapping for each block's data into the larger data structures. Therefore, the block extraction can occur within the $for - loop$ in line 8 of algorithm 1 independent of any other block b. The same applies to the separation set blocks. Further, in order to achieve the overlap, we utilize CUDA streams. Hence, a block b is transferred on one CUDA stream, while in a second CUDA stream a kernel is executed on another block b at the same time. Once the kernel execution is finished, data is transferred back to the host and merged on the CPU at the end of the $for - loop$, see line 21 of algorithm 1.

3 Experimental Results

In a first experiment, we determine the best performing block size *bs*, based on an artificial dataset with 20 000 variables. We assume a worst case, in which all CI tests have to be conducted within levels $l = 0, 1$. We report the average execution time of 10 runs in table 1. The results indicate that larger block sizes provide better execution times for the block-wise adjacency search. In particular, a block size $bs = 4096$ shows the best performance. The largest block size measured $bs = 8192$ and small block sizes $bs = 64, 128$ show poor performance. For the larger block size, we assume that its dimension is not suitable for the dataset as only half of the last block is utilized. Using the profiler `nvprof`, we found out that the kernel executions for the smaller block sizes result in a low `sm_efficiency` and the CPU computations and data transfer cannot fully overlap, with GPU kernel execution.

Table 1: Average execution times *time* of 10 runs of the block-wise adjacency search, processing a dataset with 20 000 variables conducting all CI tests within levels $l = 0, 1$ with varying block sizes *bs*

bs	64	128	256	512	1024	2048	4096	8192
time in *s*	6042.6	5072.6	4909.5	4935.7	4862.1	4826.3	4788.2	5561.9

In a second experiment, we measure the average execution time of 10 runs of the block-wise implementation with different suitable block sizes *bs* and increase the number of variables in the datasets. The results are shown in figure 2. First, the `noBlocks` version represents the baseline GPU-accelerated implementation from previous work [10], which does not support processing of datasets exceeding the device memory capacity. On the available K80 GPUs, processing of datasets above 20 000 variables fails. In contrast, the block-wise GPU-accelerated adjacency search is able to process datasets beyond the device memory limitation, i.e., datasets with up to 45, 000 variables. Also, there is no additional performance degradation visible, when datasets are processed, which exceed the GPU memory capacity. Second, comparing the different block sizes *bs* confirms the results from the first experiment. While, the performance gap is small, the larger block size $bs = 4096$ leads to faster execution times. Given the memory constraint for the block size *bs*, which restricts the available memory for each block *b* for higher level $l \geq 1$, smaller block sizes may be required.

Note, recent GPU-generations include a page migration engine that transfers data that is required within GPU kernels from host to device memory transparently. Thereby, allowing to oversubscribe the available GPU memory. Yet, experiments show that the block-wise version remains faster, by avoiding any page faults and overlapping data transfer with compute [9].

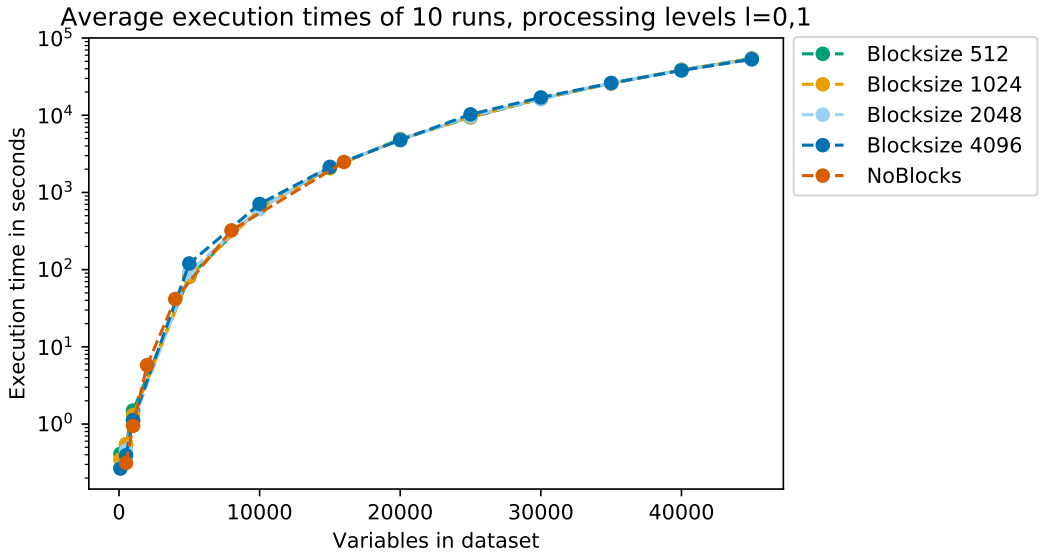


Figure 2: Average execution times (10 runs) of the block-wise (with different block sizes *bs*) and non-block-wise (noBlocks) adjacency search for levels $l = 0, 1$ on the GPU, varying the number of variables in the datasets. Note, all CI tests are conducted within each level.

4 Summary

Using the K80 GPUs provided by the Future SOC Lab, we are able to experimentally evaluate an extension of an existing GPU-accelerated adjacency search used in constraint-based causal structure learning algorithms. The extension, called block-wise adjacency search, splits input datasets into smaller blocks to enable processing of high-dimensional datasets, e.g., as present in genetic research, that exceed the GPU memory capacity. We demonstrated experimentally that the chosen block size *bs* has an impact on the execution time. For smaller block sizes CPU computations and data transfer do not fully overlap with GPU kernel execution. Larger block sizes may not be suitable for higher level $l \geq 1$. Yet, an experimental investigation of the behaviour for higher level $l \geq 1$ is still open. Further, the block-wise adjacency search enables processing on multiple GPUs, which we do not cover in this report.

References

[1] Cancer Genome Atlas Research Network, J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, and J. M. Stuart. "The Cancer Genome Atlas Pan-Cancer analysis project". In: *Nat Genet* 45.10 (Oct. 2013), pages 1113–1120.

[2] D. Colombo and M. H. Maathuis. "Order-independent Constraint-based Causal Structure Learning". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pages 3741–3782. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2750365.

[3] A. P. Dawid. "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pages 1–31. ISSN: 00359246. URL: http://www.jstor.org/stable/2984718.

[4] M. Kalisch and P. Bühlmann. "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm". In: *J. Mach. Learn. Res.* 8 (May 2007), pages 613–636. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1248659.1248681.

[5] NVIDIA Corporation. *NVIDIA Tesla K80 The World's fastest GPU Accelerator*. Nov. 2014.

[6] J. Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 978-0-521-89560-6.

[7] C. Perscheid, B. Grasnick, and M. Uflacker. "Integrative Gene Selection on Gene Expression Data: Providing Biological Context to Traditional Approaches". In: *Journal of Integrative Bioinformatics* (2018). Edited by F. Schreiber and R. Hofestädt.

[8] A. Rau, F. Jaffrézic, and G. Nuel. "Joint estimation of causal effects from observational and intervention gene expression data". In: *BMC systems biology* 7.1 (Oct. 2013), page 111. DOI: 10.1186/1752-0509-7-111.

[9] C. Schmidt, J. Huegle, S. Horschig, and M. Uflacker. "Out-of-Core GPU-Accelerated Causal Structure Learning". In: *19th International Conference on Algorithms and Architectures for Parallel Processing, Melbourne, Australia | 9-11 Dec 2019*. 2019, to appear.

[10] C. Schmidt, J. Huegle, and M. Uflacker. "Order-independent Constraint-based Causal Structure Learning for Gaussian Distribution Models Using GPUs". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. SSDBM '18. Bozen-Bolzano, Italy: ACM, 2018, 19:1–19:10. ISBN: 978-1-4503-6505-5. DOI: 10.1145/3221269.3221292. URL: http://doi.acm.org/10.1145/3221269.3221292.

[11] M. Scutari. "Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package". In: *Journal of Statistical Software, Articles* 77.2 (2017), pages 1–20. ISSN: 1548-7660.

[12] P. Spirtes. "Introduction to Causal Inference". In: *J. Mach. Learn. Res.* 11 (Aug. 2010), pages 1643–1662. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1859905.

[13] B. Zarebavani, F. Jafarinejad, M. Hashemi, and S. Salehkaleybar. "cuPC: CUDA-based Parallel PC Algorithm for Causal Structure Learning on GPU". In: *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2019). DOI: 10.1109/TPDS.2019.2939126.

Process-based machine learning to analyse HEI compliance

Ildikó Szabó, Szabina Fodor, and Katalin Ternai

Department of Information Systems
Corvinus University of Budapest
ildiko.szabo2@uni-corvinus.hu, szabina.fodor@uni-corvinus.hu, katalin.ternai@uni-corvinus.hu

Digitalization transforms not just business processes and models of companies, but cognitive processes of their employees. The usage and development of new technologies boosting the fourth industrial revolution will require specific competences, meanwhile outdated competences will be eliminated after a while. Educational institutions must prepare themselves for these changes. This paper presents a data warehouse approach to monitor actual labour market needs and forecast future competences.

1 Introduction

Employees' knowledge are important elements of running, monitoring and restructuring business processes. Knowledge are also required to develop, apply or maintain technological innovations. In the era of fourth revolution, educational institutions have a responsibility to educate the future employees. They need predictions about future competences to start modifying, updating curricula in time, because students take 3-5 years to be graduated. Predictions of competences are created based on experts' opinions [1, 3, 4, 9], but the causal relationships are not formalized in these cases. Models and statistical analyses can discover these relationships behind the data. Statistical analyses require time series data to detect trend and so on. Data warehouses can consolidate these subject-oriented, non-volatile and integrated time series data. Job ads contain competences needed by a labour market per region, time and occupation. ETL process extracts, cleansed and load them from streaming data of job portals into the DW. Business analytics facilitates to analyse trends. These trends are used to check the compliance of HEI educational portfolios and their learning outcomes with future competence needs. Learning analytics helps to analyse student learning processes to make their knowledge acquisition activities more improved.

Contributions

Our research aims at creating a data warehouse to assess future job competences collecting time series data from job portals and transforms them into the data warehouse. It requires to develop a process-based machine learning algorithm to extract competence patterns from job descriptions.

2 Context

Data derived from different data sources are consolidated in a data warehouse to provide business enterprise view of all information. Data warehouse contains connected data tables determined by business analytical purposes. Dimension tables represent qualitative data, fact tables store mainly quantitative data. These tables can be ordered a star schema or snowflake schema to represent multidimensionality and provide opportunities for analysing data along different dimension or their hierarchy [2]. Turban at al. [8] distinguished four types of architectures. Data mart is a small data warehouse to meet departmental requirements instead of enterprise wide ones. Virtual, distributed and federated type is not a data warehouse in the traditional sense. A middleware layer is created to handle, organize and transform data from different data sources for providing analytical capabilities. Hub-and-Spoke and Enterprise DW are to define an enterprise-centric approach including common data definitions, formats for managing business in a consolidated way. However, data marts are obtained from the data warehouse in the first case to facilitate departmental decision making, but this results redundant data and their maintenance cost. ETL (Extract, Transformation, Load) process is required to provide a consolidated, enterprise wide view of business by data warehouses. During this process data are extracted from different sources (operational systems, flat files etc.), transformed into a predefined, agreed data structure and loaded into the data warehouse [2].

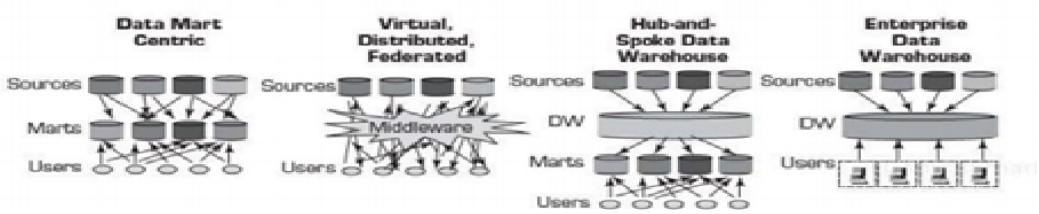


Figure 1: DW Architectures

3 Problem

Our purpose is to create a data warehouse to analyse time series and regional data about competences for monitoring labour market needs and detecting changes caused by technological innovations boosting the fourth industrial revolution.

4 Solution

Approximately 1000 job advertisements were downloaded from Indeed[1] job portal in September 2019. Job vacancies are from manufacturing industry. They contain data about country or city where competence needs emerged, company and occupation requiring these competences, job description contain information about them, date when these needs appeared on the web and salary. Collected data will be organized into the fol-lowing dimension and fact tables: Dimension tables are the following ones, every tables have primary key not mentioned here.

- Location table contains Location ID, Country, Region and City data.
- Company table includes Company ID, Occupation, Company, its related Industry and Industrial code from a national framework (like NACE, SOC etc.) .
- Competence table has Competence ID, Professional, Personal, Social and Methodological Competence fields.
- Date table contains data of publishing date (year, quarter, and month) and all of them are part of the key field.

The related fact table includes data about salary and IDs of these dimension tables. To create a data warehouse the following problems must be solved by the algorithms of the ETL process:

- Job portals do not publish region and city data separately, hence they have to be separated.
- Names of occupations are within a wide range scale, they have to be classified.
- Competences have to be extracted from job descriptions and organized into one of the four categories.

The ETL process is under development. This paper presents only the competence extraction process.

5 Implementation

Streaming data coming from job portals can be stored in SAP Hana running on the HPI Future SOC Lab architecture. SAP Hana Text Analytics can be used to implement developed process-based machine-learning algorithm. This algorithm uses process ontologies transformed from business process modelling and semantics of relationships between process model elements as open queries. First version of this algorithm was presented in [5, 6, 7]. The first phase is to create an ADONIS process

[1] https://www.indeed.com (last accessed 2020-04-01).

model from a selected manufacturing process and transform it into the Reference Process Ontology (RPO) with using a JAVA transformation program. ADONIS is a graph-structured Business Process Management language. Its main feature is the method independence. Our approach is principally transferable to other semi-formal modelling languages. The semantic annotation for specifying the semantics of the tasks and decisions in the process flow explicitly is important in our method. The second phase is to discover new process elements within job descriptions with the help of semantic text mining. The algorithm focuses on finding patterns shaped into open queries. Relations are regarded as ordered pairs. The algorithm assumes that certain expressions can represent a given relation within the document e.g. produces output(Process step, Document). A relation suggests that something must be happen with a document e.g. it is submitted or signed. That's why the algorithm wants to find x submit y pattern within the document, where y is a document. It seeks „submit" term and collects few words after that.

6 Conclusion

This paper presented a solution how data about labour market needs can be consolidated into a data warehouse. There are other solutions to analyse job market needs. But they are mainly to match job seekers profiles with actual labour market requirements and not to store these data for analysing trends of competence needs. These analyses require unified, integrated time series data in the meaning of their syntax and semantic as well which can be provided by a data warehouse.

References

[1] Cedefop. *Briefing note - People, machines, robots and skills*. July 2017. URL: http://www.cedefop.europa.eu/en/publications-and-resources/publications/9121 (last accessed 2020-04-01).

[2] S. Chaudhuri and U. Dayal. "An Overview of Data Warehousing and OLAP Technology". In: *SIGMOD Rec.* 26.1 (Mar. 1997), pages 65–74. ISSN: 0163-5808. DOI: 10.1145/248603.248616.

[3] A. Davies, D. Fidler, and M. Gorbis. "Future work skills 2020". In: *Institute for the Future for the University of Phoenix Research Center* (2011).

[4] OECD. *Skill for a Digital World*. URL: http://www.oecd.org/els/emp/Skills-for-a-Digital-World.pdf (last accessed 2020-04-01).

[5] I. Szabó and K. Ternai. "Process-Based Analysis of Digitally Transforming Skills". In: *Research and Practical Issues of Enterprise Information Systems*. Edited by A. M. Tjoa, L.-R. Zheng, Z. Zou, M. Raffai, L. D. Xu, and N. M. Novak. Lecture Notes in Business Information Processing. Cham: Springer International

Publishing, 2018, pages 104–115. ISBN: 978-3-319-94845-4. DOI: 10.1007/978-3-319-94845-4_10.

[6] I. Szabó and K. Ternai. "Semantic Audit Application for Analyzing Business Processes". In: *Research and Practical Issues of Enterprise Information Systems: 10th IFIP WG 8.9 Working Conference, CONFENIS 2016, Vienna, Austria, December 13–14, 2016, Proceedings*. Edited by A. M. Tjoa, L. D. Xu, M. Raffai, and N. M. Novak. Cham: Springer International Publishing, 2016, pages 3–15. ISBN: 978-3-319-49944-4. DOI: 10.1007/978-3-319-49944-4_1.

[7] K. Ternai and I. Szabó. "Semantic Audit Application." In: *International Semantic Web Conference (Posters & Demos)*. 2016.

[8] E. Turban, R. Sharda, and D. Delen. *Decision support and business intelligence system*. 10th edition. Upper Saddle River, New Jersey: Pearson Pentice Hall, 2007. ISBN: 978-0-13-610729-3.

[9] World Economic Forum. *The jobs of the future – and two skills you need to get them*. Sept. 2016. URL: https://www.weforum.org/agenda/2016/09/jobs-of-future-and-skills-you-need/ (last accessed 2020-04-01).

Improving Test Suite Generation by Testing Google Play's Top 1000 Apps III

Thomas Vogel[1], Chinh Tran[2], Irene Moser[3], and Lars Grunske[1]

[1] Software Engineering Group
Humboldt-Universität zu Berlin, Germany
thomas.vogel@informatik.hu-berlin.de
grunske@informatik.hu-berlin.de
[2] Technische Universität Berlin, Germany
chinh.tran@mail.de
[3] Swinburne University of Technology, Australia
imoser@swin.edu.au

Software testing plays an important role in providing evidence for the correctness of software systems. Due to the costs and complexity of testing such systems, tests are often automatically generated using search-based techniques. The goal of this project is to improve the generation of test suites for mobile applications by adapting existing search techniques based on fitness landscape analyses. Similarly to the previous reports of the first two project phases [10, 11], we motivate and discuss the research context of the project, and present the results from the third project phase in the FutureSOC lab. We also summarize the results from all three phases.

1 Motivation and Research Context

To provide evidence for the correctness of software systems, testing plays an important role, in practice [2, 6] as well as in software engineering research where it is an active topic of research. In the context of search-based software engineering [4], researchers investigate the automated, search-based generation of test data and test suites because manual testing of software systems is complex and costly [3, 7].

One application area of automated, search-based generation of test suites are mobile applications (apps) [1, 6]. In this area, one approach is *Sapienz* that has discovered bugs (in terms of crashes) in roughly every third Android app that has been tested [6]. This illustrates the relevance of testing and the need to continuously improve test generation approaches.

In search-based testing, tests are automatically generated by heuristics that search for optimal tests guided by a fitness function. The heuristic or generally search technique (e.g., a genetic algorithm) that is used in an approach is often selected and configured in black-box fashion because the characteristics of the search problem are unknown. Consequently, the expected performance of a search technique is also unknown. In practice, two possible options of selecting and configuring a search technique are typically used: (1) try out different techniques to identify which one works best for the given problem, or (2) select a popular technique and use its default

configuration. Option (1) has the drawback of being costly and time-consuming, and option (2) does not guarantee that the selected technique works well on different specific problems such as different apps (*cf.* no-free-lunch theorems for search [12, 13]). This requires selecting and configuring search techniques based on the specific search problem, among others, given by the app under test.

For this purpose, we aim for a better understanding of the search problem by analyzing the fitness landscape [5, 8]. These analysis results will provide feedback for selecting and especially configuring the search technique for a test generation approach so that better tests are generated, for instance, with respect to achieved coverage and number of revealed faults. Particularly, we consider the application area of generating test suites for Android apps and aim for improving the state-of-the-art approach Sapienz [6]. Therefore, we plan a large study of testing the top 1000 apps of the Google Play store. This study will further provide a report about the state of the practice in how reliable today's most popular apps are.

The interested reader is referred to [10] for a refined discussion of the research problem and goals of this project.

2 Experiments and Results

Based on a fitness landscape analysis of generating test suites for apps, we observed that the diversity of test suites generated by Sapienz decrease over time during the search. Since diversity is a critical factor for the success of a search, we proposed Sapienz^{div} as an adaptation of the search technique of Sapienz that includes four mechanisms that maintain the diversity during the search [9]. In the FutureSOC lab, we evaluate Sapienz^{div} in a head-to-head comparison with Sapienz.

Project Phase I – Deployment and Distribution Model: In the first project phase, we determined the deployment and distribution model for the evaluation. Particularly, each server runs one instance of Sapienz/Sapienz^{div} to test one app, and uses ten Android emulators to execute ten test suites concurrently on this app during the search. Thus, we use servers to run Sapienz/Sapienz^{div} concurrently for different apps and the resources of one server to parallelize the execution of test suites for an individual app. Details can be found in an earlier report [10].

Project Phase II – Experiment with the 68 App Benchmark: For the first experiment, we use the 68 app benchmark proposed by Choudhary *et al.* [1] consisting of 68 open-source apps from F-Droid. We run Sapienz and Sapienz^{div} once on each app over ten generations of search, and measure the coverage achieved by the generated test suites, the number of identified faults (crashes), the length of the test sequences, and the execution time of the search. The results indicate that Sapienz^{div} and Sapienz achieve similar statement coverage (on average 45.67 vs. 45.05), and that Sapienz^{div} can outperform Sapienz in revealing crashes (in total 141 vs. 119 unique crashes) while producing longer fault-revealing test sequences (on average

244 vs. 209 events) and requiring more time for the search (on average 118 vs. 101 min). Details can be found in an earlier report [11] and the research paper [9].

Project Phase III – Experiment with Inferential Statistical Analysis: In the current phase of the project, we use ten further open-source F-Droid apps from [6] and run Sapienz respectively Sapienz^{div} 20 times on each of these apps for ten generations of search. Results show that Sapienz significantly outperforms Sapienz^{div} with large effect size on all apps for execution time. Concerning the other aspects, the results are inconclusive. Sapienz^{div} significantly outperforms Sapienz with large effect size on only 3/10 apps for coverage, 2/10 for the number of crashes, and almost 1/10 for the length of test sequences. The remaining results are not statistically significant or do not indicate large differences. Thus, the results are inconclusive in differentiating both approaches by their performance (*cf.* [9]) and thus, they do not confirm the differences in fault revelation capabilities on the 68 app benchmark.

Given these results, we are currently conducting experiments over 40 generations of search to investigate the effect of the diversity-preserving mechanisms of Sapienz^{div} when the lack of diversity becomes clearly noticeable in Sapienz and the search of Sapienz stagnates (after 25 generations). While one run of a search over ten generations takes around 120 min for one app, a search run over 40 generations will consume considerably more time.

3 Used HPI Future SOC Lab Resources

As before [11], the experiments were performed on eight blade servers (*hum-nodes*) of the HPI Future SOC Lab. To parallelize the execution, at any point in time during an experiment one app was tested on each server. For each server, the evaluations of test suites (i.e., the execution of multiple test suites) for one app was parallelized by running ten Android emulators concurrently (*cf.* deployment and distribution model [10]).

4 Conclusion and Next Steps

In this report, we have discussed the research context (*cf.* [10, 11]), our experiments conducted so far, and the results. The research results have been published in [9]. They indicate that Sapienz^{div} does not improve Sapienz in a statistically significant way for a search over ten generations. Therefore, we are currently extending the number of generations to 40 to investigate the performance of Sapienz^{div} when the search of Sapienz actually stagnates (after 25 generations).

Given the estimated costs of testing one app over 40 generations, we cannot fulfill our initial project plan of testing Google Play's top 1000 apps in this FutureSOC lab period. On the other hand, testing these apps over fewer generations (e.g., ten) will likely not show any statistically significant difference between Sapienz and Sapienz^{div}

(*cf.* experiments conducted so far). Therefore, testing Google Play's top 1000 apps is left for future work.

References

[1] S. R. Choudhary, A. Gorla, and A. Orso. "Automated Test Input Generation for Android: Are We There Yet? (E)". In: *30th International Conference on Automated Software Engineering*. ASE '15. IEEE, 2015, pages 429–440.

[2] G. Fraser and A. Arcuri. "1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite". In: *Empirical Software Engineering* 20.3 (2015), pages 611–639.

[3] M. Harman. "The Current State and Future of Search Based Software Engineering". In: *Workshop on the Future of Software Engineering, FOSE*. 2007, pages 342–357.

[4] M. Harman, S. A. Mansouri, and Y. Zhang. "Search-based Software Engineering: Trends, Techniques and Applications". In: *ACM Comput. Surv.* 45.1 (2012), 11:1–11:61.

[5] K. M. Malan and A. P. Engelbrecht. "A survey of techniques for characterising fitness landscapes and some possible ways forward". In: *Information Sciences* 241.Suppl. C (2013), pages 148–163.

[6] K. Mao, M. Harman, and Y. Jia. "Sapienz: Multi-objective Automated Testing for Android Applications". In: *25th International Symposium on Software Testing and Analysis*. ISSTA '16. ACM, 2016, pages 94–105.

[7] P. McMinn. "Search-based Software Test Data Generation: A Survey". In: *Software Testing, Verification and Reliability* 14.2 (2004), pages 105–156.

[8] E. Pitzer and M. Affenzeller. "A Comprehensive Survey on Fitness Landscape Analysis". In: *Recent Advances in Intelligent Engineering Systems*. Springer, 2012, pages 161–191.

[9] T. Vogel, C. Tran, and L. Grunske. "Does Diversity Improve the Test Suite Generation for Mobile Applications?" In: *11th International Symposium on Search-Based Software Engineering*. SSBSE '19. Springer, 2019, pages 58–74.

[10] T. Vogel, C. Tran, I. Moser, and L. Grunske. *Improving Test Suite Generation by Testing Google Play's Top 1000 Apps*. Technical report. HPI Future SOC Lab Project Reports, 2018.

[11] T. Vogel, C. Tran, I. Moser, and L. Grunske. *Improving Test Suite Generation by Testing Google Play's Top 1000 Apps II*. Technical report. HPI Future SOC Lab Project Reports, 2019.

[12] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Trans. on Evolutionary Computation* 1.1 (1997), pages 67–82.

[13] D. H. Wolpert and W. G. Macready. *No Free Lunch Theorems for Search*. Technical report SFI-TR-95-02-010. Santa Fe Institute, 1995.

Deep Representation Learning on Large Attributed Graphs

Debora Nozza[1] and Enza Messina[1]

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano Bicocca
{debora.nozza,messina}@unimib.it

Attributed graphs convey a rich set of information on relations, nodes, and attributes. However, the entire set of information cannot be easily captured by traditional Representation Learning methods usually adopted for creating low-dimension and meaningful embeddings of the graph. This research project aims to generate a representation that encodes both the relational structure of a graph and the node attributes that can be as texts, images. The final goal is to obtain an unsupervised model based on promising Deep Learning architectures that is able to efficiently and effectively derive dense vector representations of nodes in attributed graphs.

1 Introduction

One of the most crucial phase when addressing a machine learning problem is the definition of data representation. The input representation can considerably influence the performance of machine learning models, depending on its ability to disentangle and discover explanatory factors of variations behind the data given as input. A common practice is to exploit a-priori knowledge in order to design an ad-hoc representation depending on the domain, task or application. Although this feature engineering usually leads to improved results, it has several drawbacks: it requires the need of a domain expert resulting in a labor-intensive and not generalizable effort and it considers the data as identically distributed (i.i.d. assumption).

Representation Learning has become an important research field with the aim of providing novel methods for learning representation of the data that make it easier to extract useful information when building classifiers [15]. The interest in this field has recently grown because of the advent of Deep Learning and the ability to deal with unsupervised and semi-supervised learning.

The majority of the research contributions on Deep Learning focus on efficiently learning good representation for i.i.d. data [7, 9]. However, data can be represented in various forms and, in particular, relational structures are common representations used in many real-world problems, e.g. airline networks, publication networks, social and communication networks, and the World Wide Web.

Dealing with relational structures, such as graphs, is very complex and computationally expensive because of different characteristics of the data, i.e. size, dynamic nature, noise, and heterogeneity [3]. One efficient approach for handling potentially large and complex graph is to learn the graph representations, or Graph Embeddings [2, 4], which assign to each node of the graph a low-dimensional dense vector

representation, encoding meaningful information conveyed by the graph. Once the embedded representation is obtained, a wide variety of data mining problems can be solved by applying off-the-shelf algorithms designed for handling vector representations.

Several graph representation learning approaches at the state of the art focus only on the graph structure to compute the graph embeddings [10, 12, 14]. However, nodes in real-world graphs are often associated with a rich set of features or attributes (e.g. text, image, audio), originating the so-called attributed graph.

Although attributed graphs convey a rich set of information on relations, nodes, and attributes, they cannot be easily given as input to classic representation learning methods. In order to overcome this limitation, the main goal of this project is the generation of a representation able to encode both the node attributes and the graph structure. Indeed, the consideration of both information could strongly improve the meaning encoded in the representation and consequently the performance of the graph mining tasks.

2 Problem

The research project aims at designing and developing novel unsupervised models for learning a graph representation from large heterogeneous attributed graphs, which comprises both the structure of the graph and the attributes associated with each node. The proposed graph representation learning model will be based on deep learning models, strengthened by an efficient optimization algorithm able to scale for large graphs.

The proposed research project will face the following challenges:

1. Structure-preserving: graph embeddings should preserve the structure of the graph, which is often complex and highly non-linear. Moreover, how to simultaneously preserve the local and global structure is also a tough problem.

2. Scalability: most real-world graphs are huge and contain millions of nodes and edges. The graph representation learning model should be scalable and able to process large graphs.

3. Sparsity: many real-world graphs are often so sparse that considering only (few) observed links is not enough to reach a satisfactory performance [10].

4. Dimensionality of the embedding: the dimension of the embedded representation should be chosen as a trade-off between reconstruction precision performances and time and space complexity. The choice can also be application-specific depending on the objective task.

5. Attribute expression: the obtained graph embedding should be able to directly encode the attribute features in addition to the graph structure information.

3 Implementation

The research project will be implemented in Python, by taking advantage of the Keras[1] library for Deep Learning. Keras is a minimalist, highly modular neural networks library written in Python and capable of running on top of either TensorFlow or Theano. The Keras Deep Learning library permits to:

- Easy and fast prototype, through user friendliness, modularity, and extensibility.
- Support several Deep Learning architectures, such as convolutional and recurrent networks.
- Run code on CPU and GPU.

Since large attributed graphs, which is the input structure that we want to investigate, are commonly associated with millions of nodes and related attributes [13], they require a large amount of memory to be loaded. Moreover, the use a GPUs would permit to substantially scale and more efficiently deal with the training process of Deep Learning models on large amount of data. We gratefully acknowledge the support of the HPI Future SOC Lab, for providing the IT infrastructure and the access to NVIDIA Tesla K80 that permits the realization of this project.

4 Evaluation

The evaluation has been performed on the task of node classification on the first connected component of the citation network benchmark DBLP dataset. A citation network is a graph where the nodes are composed of a set of research articles, the node attribute is the title and the edges are the citation relationships between articles.

Table 1: Comparison of a model that consider both attributes and relational information vs state of the art models that consider only relational information on DBLP using 50 % of labeled data

	Attributes + Relations	**SDNE**[14]	**node2vec**[5]	**DeepWalk**[10]
Accuracy	**0.7479 ± 0.004**	0.4684 ± 0.003	0.6933 ± 0.004	0.4952 ± 0.000
F-measure$_{macro}$	**0.6831 ± 0.005**	0.2010 ± 0.002	0.5824 ± 0.005	0.2841 ± 0.000

From the table, it clearly emerges that considering both the textual attributes and the interactions of the nodes provides significant improvements compared to the results obtained by modeling only the relational information.

[1] https://keras.io/ (last accessed 2020-04-01).

5 Related Work

Over the past decade, graph representation learning has attracted a surge of research attention, particularly focused on developing new embedding algorithms. In this domain, research studies can be roughly distinguished in factorization methods and deep learning approaches. The basic idea underlying both methods is to preserve both the local and global graph structure in the embedded vector space.

Factorization based algorithms represent the connections between nodes in the form of a matrix and factorize this matrix to obtain the embeddings [1, 2, 8, 11, 12]. The growing interest in deep learning algorithms has affected also the task of graph representation learning due to their ability to model non-linear structures in the data. Beyond Deep Learning methods applied on random walks [10], most of the investigations are focused on improving deep learning architectures [14].

Only a few studies considered the set of features associated to each node in addition to the graph topological structure (attributed graph) [3, 6]. Their limitation regards the fact that they do not explicitly consider the nodes' attributes but only a measure of attribute similarity between them, resulting in a lower representation expressiveness.

6 Conclusion

The research project will aim to propose a novel unsupervised model for attributed graph embeddings. It has been preliminary demonstrated that considering the relational information in addition to the attribute information will provide significant improvements.

Future work will be focused on dealing with attributed graphs with probabilistic relationships. Creating meaningful embeddings of attributed graphs where noisy and uncertain relations are considered represents a major challenge for tackling real word problems.

References

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. "Distributed Large-scale Natural Graph Factorization". In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pages 37–48.

[2] M. Belkin and P. Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". In: *Neural Computation* 15.6 (2003), pages 1373–1396.

[3] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. "Heterogeneous Network Embedding via Deep Architectures". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pages 119–128.

[4] P. Goyal and E. Ferrara. "Graph Embedding Techniques, Applications, and Performance: A Survey". In: *Knowledge-Based Systems* 151 (2018), pages 78–94. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.03.022. arXiv: 1705.02801 [cs.SI].

[5] A. Grover and J. Leskovec. "Node2Vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pages 855–864.

[6] X. Huang, J. Li, and X. Hu. "Accelerated Attributed Network Embedding". In: *Proceedings of the 2017 SIAM International Conference on Data Mining*. 2017, pages 633–641.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* (2013). arXiv: 1301.3781 [cs.CL].

[8] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. "Asymmetric Transitivity Preserving Graph Embedding". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pages 1105–1114.

[9] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Volume 14. 2014, pages 1532–1543.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena. "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pages 701–710.

[11] S. T. Roweis and L. K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500 (2000), pages 2323–2326.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. "LINE: Large-scale Information Network Embedding". In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pages 1067–1077.

[13] L. Tang and H. Liu. "Scalable learning of collective behavior based on sparse social dimensions". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pages 1107–1116.

[14] D. Wang, P. Cui, and W. Zhu. "Structural Deep Network Embedding". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pages 1225–1234.

[15] A. C. Yoshua Bengio and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* 35.8 (2013), pages 1798–1828.

Learning to Generate from Fact Representations

Rajarshi Bhowmik and Gerard de Melo

Deep Data Lab
Rutgers University–New Brunswick, NJ, USA
rajarshi.bhowmik@rutgers.edu, gdm@demelo.org

In recent years, there has been substantial progress on powerful machine learning techniques that generate sequences as output. In this work, we present an approach to more flexibly generate natural language text from heterogeneous data such as as facts coming from a database. We induce representations for these facts and allow the model to flexibly incorporate information from these facts into the generated output. Our experiments show that our method outperforms previous approaches.

1 Introduction

Motivation While most machine learning approaches seek to predict an individual output value such as a number or a simple classification label, there has also been research on structured prediction approaches that produce entire sequences as output. These are particularly useful when we wish to learn a model that generates natural language text as output. In recent years, there has been substantial progress on such techniques. Important breakthroughs include RNN and LSTM-based neural sequence-to-sequence models [8], as well as sequence-to-sequence models with a soft attention mechanism [2]. These models assume that the input is a sequence just like the output. However, in many cases we wish to generate a sequence from inputs that are of a different form. In particular, we may have structured data coming from a database as input, and may wish to learn how to generate natural language text from these.

Project Idea In our work, we propose a model that produces a sequence as output, but is able to consider structured data as input. Specifically, we consider a model that can take a series of facts as input, and generate natural language text as output, while flexibly incorporating items appearing in the facts into the output. By inducing representations of facts and being able to incorporate them into the output, the model is flexibly able to incorporate arbitrary previously unseen names on demand. For example, the machine learning algorithm may never have seen the string *Havel* in its training data. However, by inducing a fact representation that encodes the entity *Havel*, the algorithm is nevertheless able to produce text as output that contains the name *Havel*. For this, the model needs to be able to refer to specific items in the input facts and must be able to choose to incorporate them into the output directly.

2 Method

2.1 Fact Representation Induction

Given input facts $\mathcal{F} = \{f_1, f_2, \ldots f_N\}$, each of which is a property–value pair, we use the positional encoding scheme [3, 7] that converts each f_i into a vector $\mathbf{f}_i = \sum_j \mathbf{l}_j \odot \mathbf{w}_j^i$. Here, $\odot$ denotes element-wise multiplication, $\mathbf{l}_j$ is a column vector with the structure $l_{kj} = (1 - \frac{j}{J}) - (k/d)(1 - 2\frac{j}{J})$, with J being the number of words in the fact, and $\mathbf{w}_j^i \in \mathrm{R}^d$ is the word embedding of the j-th word in fact f_i (we concatenate property name and value for this, and out-of-vocabulary words are replaced with the special token <*UNK*>). Finally, we append a special *mean fact representation* $\mathbf{f}_{N+1} = \frac{1}{N}\sum \mathbf{f}_i$ and later train the model to attend to this mean fact whenever it has to generate a vocabulary word rather than a fact word.

2.2 Output Generation

At every time step t, the generator selects a fact and then generates the next output element either from the fact representations, or from the default vocabulary.

Fact Selection The selection is achieved using an attention mechanism. Attention scores are computed as a likelihood distribution using the hidden state of the generator in the previous time step and the fact embeddings. Formally,

$$\mathbf{e}_i = \mathbf{W}_2 \tanh(\mathbf{W}_1[\mathbf{f}_i; \mathbf{h}_{t-1}]) \;\; \forall i \in \{1, N+1\}, \tag{1}$$

$$P(f = f_i \mid \mathbf{f}_i, \mathbf{h}_{t-1}) = \frac{\exp(\mathbf{e}_i)}{\sum_{i' \in \{1,N+1\}} \exp(\mathbf{e}_{i'})}, \tag{2}$$

where $\mathbf{e}_i$ denotes the attention energy of the i-th fact, $[;]$ denotes concatenation, and $\mathbf{W}_1 \in \mathrm{R}^{m \times 2d}$, $\mathbf{W}_2 \in \mathrm{R}^m$ are affine transformations of a 2-layer feed-forward network.

We select the fact with maximum attention score at time step t, denoted as f_t and its corresponding fact embedding $\mathbf{f}_t$ as

$$f_t = \underset{i \in \{1,\ldots,N+1\}}{\arg\max} \; P(f = f_i \mid \mathbf{f}_i, \mathbf{h}_{t-1}) \tag{3}$$

$$\mathbf{f}_t = \mathbf{f}_{f_t}. \tag{4}$$

Decoder We then create a concatenation $\mathbf{x}_t = [\mathbf{f}_t; \mathbf{w}_{t-1}; \mathbf{v}_{t-1}]$ of three vectors: the embedding $\mathbf{f}_t$ of the fact selected for the current time step, the embedding $\mathbf{w}_{t-1}$ of the vocabulary word at the previous time step, and a one-hot vector $\mathbf{v}_{t-1}$ corresponding to the position of the copied factual word in the previous time step. Since the generated word in the previous time step can either be a vocabulary word or a factual word, either $\mathbf{w}_{t-1}$ or $\mathbf{v}_{t-1}$ is set to a zero vector.

The concatenation is fed to a GRU decoder, which updates its state as $\mathbf{h}_t = \mathrm{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1})$. If the attention mechanism assigns the maximum score to the mean fact f_{N+1}, the decoder generates a vocabulary word $w_t \in \mathcal{V}$. For this, we use the

attention-weighted context $\mathbf{c}_t = \sum_i \alpha_i \mathbf{f}_i$ and the GRU output state $\mathbf{h}_t$. A concatenation of these two vectors is fed to a 2-layer feed-forward network with a non-linear ReLU activation applied to the hidden layer as $\mathbf{o}_t = \mathbf{W}_a \text{ReLU}(\mathbf{W}_b[\mathbf{c}_t; \mathbf{h}_t])$, where $\mathbf{W}_a$ and $\mathbf{W}_b$ are affine transformations and $\text{ReLU}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$. Finally, a probability distribution over the vocabulary words is obtained by a softmax function and the word with the maximum probability is emitted by the decoder:

$$P(w \mid \mathbf{c}_t, \mathbf{h}_t) = \text{Softmax}(\mathbf{o}_t) \tag{5}$$

$$w_t = \arg\max_{w \in \mathcal{V}} P(w \mid \mathbf{c}_t, \mathbf{h}_t). \tag{6}$$

If, in contrast, if the fact selection process chooses one of the N facts, the decoder copies words from it directly to the output. For this, the decoder must select the position index of the factual word within the factual phrase. The index is predicted by a 2-layer feed-forward network that takes a concatenation of the selected fact embedding $\mathbf{f}_t$ and the output state of the GRU $\mathbf{h}_t$ as input. Then, the position index of the word to copy is determined as follows.

$$\mathbf{r}_t = \mathbf{W}_c \text{ReLU}(\mathbf{W}_d[\mathbf{f}_t; \mathbf{h}_t]) \tag{7}$$

$$P(n \mid \mathbf{f}_t, \mathbf{h}_t) = \text{Softmax}(\mathbf{r}_t) \tag{8}$$

$$n_t = \arg\max_{n \in \{1, \ldots, |\mathcal{V}_{f_t}|\}} P(n \mid \mathbf{f}_t, \mathbf{h}_t) \tag{9}$$

$$w_t = (\mathcal{V}_{f_t})_{n_t} \tag{10}$$

Here, n_t is the position index of the fact word to copy, $\mathcal{V}_{f_t}$ is the sequence of fact words for f_t, and $(\mathcal{V}_{f_t})_i$ denotes the i-th item in that sequence.

2.3 Model Training

To train our model, given the input facts $\mathcal{F}$ and the fact-aligned description $\tilde{\mathcal{D}}$, the model maximizes the log-likelihood of the observed words in the description with respect to the model parameters θ as $\theta^* = \arg\max_\theta \log P(\tilde{\mathcal{D}} \mid \mathcal{F})$. This can be further decomposed as $\log P(\tilde{\mathcal{D}} \mid \mathcal{F}) = \sum_{t=1}^{|\tilde{\mathcal{D}}|} \log P(w_t \mid w_{1:t-1}, \mathcal{F})$. Since the log-likelihood of the word w_t also depends on the underlying fact selection, we can further decompose $P(w_t)$ as $P(w_t) = P(w_t \mid f_t, w_{1:t-1})\, P(f_t \mid w_{1:t-1})$. Therefore, we train our model end-to-end by optimizing the following objective function:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{|\mathcal{D}|} \log P(w_t \mid f_t, w_{1:t-1}) - \sum_{t=1}^{|\mathcal{D}|} \log P(f_t \mid w_{1:t-1}). \tag{11}$$

Note that the alignment $\tilde{\mathcal{D}}$ of the description $\mathcal{D}$ to the facts $\mathcal{F}$ can be obtained using simple heuristics rather than manual annotation.

3 Experimental Findings

We evaluate our model on the task of generating short natural language descriptions of entities on Wikidata, a large-scale knowledge graph.

Data We used randomly selected entities on Wikidata to generate two benchmark datasets, WikiFacts10k-Imbalanced and WikiFacts10k-OpenDomain, which are available from https://github.com/kingsaint/Wikidata-Descriptions.

Experimental Protocol Following previous work, we use the automatic evaluation metrics BLEU (1 - 4) [6], ROUGE-L [5], METEOR [4], and CIDEr [9] for a quantitative evaluation of the generated descriptions with respect to the ground truth descriptions provided in the benchmark data. Following standard practice, CIDEr scores are scaled to $[0, 10]$ and other scores are scaled to $[0, 100]$.

Our model and all other baselines are trained for a maximum 25 epochs. We report the results of the best performing models on the dev set. We use the 1,000 most frequent words in the dataset as our default vocabulary, and remove any generated <UNK> tokens from the output before evaluating it. Implementations of all our experiments are available online at https://github.com/kingsaint/Wikidata-Descriptions.

Table 1: Experimental results for the WikiFacts10K-Imbalanced dataset

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Dynamic Memory-based Generative Model	61.1	53.5	48.5	46.1	64.1	35.3	3.295
Fact2Seq with Attention	62.8	56.3	53.0	52.7	63.0	35.0	3.321
NKLM [1]	34.7	27.7	29.1	29.0	44.1	20.1	1.949
Our Model	**61.9**	**57.3**	**55.6**	**54.3**	**64.9**	**36.3**	**3.420**
– without Positional Encoder	58.9	51.9	46.3	42.5	63.4	33.7	3.126
– without Mean Fact	58.0	52.2	49.5	48.6	64.3	34.8	3.139
– Copy Only	26.8	21.2	16.5	11.8	45.3	20.6	1.766

Table 2: Experimental results on the WikiFacts10K-OpenDomain dataset

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Dynamic Memory-based Generative Model	57.8	50.7	43.6	39.7	67.6	34.9	3.556
Fact2Seq with Attention	62.7	56.3	50.0	46.2	67.7	35.8	3.629
NKLM [1]	47.9	42.1	37.5	32.3	57.4	25.9	2.958
Our Model	**68.2**	**61.8**	**56.6**	**51.9**	**70.0**	**37.3**	**4.084**
– without Positional Encoder	66.2	58.4	51.8	45.9	67.9	34.7	3.717
– without Mean Fact	62.4	58.0	55.1	51.8	67.7	36.0	3.856
– Copy Only	37.5	26.4	18.4	10.9	55.2	24.8	2.136

Results Tables 1 and 2 provide the evaluation results of our model and the baselines on the WikiFacts10k-Imbalanced and WikiFacts10k-OpenDomain datasets, respectively. Our model outperforms the baselines by substantial margins on both datasets.

4 Conclusion

In this report, we describe an encoder–decoder model that is flexibly able to incorporate elements from fact representations into its output. Through an extensive evaluation, we have shown that our method consistently outperforms several competitive baselines on two benchmark datasets. In terms of future work, the success of this novel architecture suggests that it could be adapted for additional applications that may benefit from a repository of structured facts, e.g. dialogue systems and knowledge graph-driven question answering.

Acknowledgments

We thank the Future SOC Lab at Hasso-Plattner-Institut for the kind provision of access to computational resources.

References

[1] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio. "A Neural Knowledge Language Model". In: *CoRR* (2016). arXiv: 1608.00318.

[2] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* (2014). arXiv: 1409.0473.

[3] R. Bhowmik and G. Melo. "Generating Fine-Grained Open Vocabulary Entity Type Descriptions". In: *Proceedings of ACL 2018*. 2018. URL: http://aclweb.org/anthology/P18-1081.

[4] A. Lavie and A. Agarwal. "Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments". In: *Proceedings of StatMT '07*. Prague, Czech Republic: Association for Computational Linguistics, 2007, pages 228–231. URL: https://aclanthology.org/W07-0734.

[5] C.-Y. Lin. "ROUGE: A Package for Automatic Evaluation of summaries". In: *Proc. ACL workshop on Text Summarization Branches Out*. 2004. URL: http://research.microsoft.com/~cyl/download/papers/WAS2004.pdf.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of ACL 2002*. 2002. URL: http://aclweb.org/anthology/P02-1040.

[7] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. "End-To-End Memory Networks". In: *Advances in Neural Information Processing Systems 28*. Edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pages 2440–2448.

[8] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. 2014. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[9] R. Vedantam, C. L. Zitnick, and D. Parikh. "CIDEr: Consensus-based Image Description Evaluation". In: *CoRR* (2014). arXiv: 1411.5726.

Benchmarking Java on Ethernet Cluster

Benchmarking PCJ – a library for parallel computing in Java

Marek Nowicki

Department of Computer Science
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University in Toruń, Poland
faramir@mat.umk.pl

The paper shows the evaluation of some tests of the PCJ library on the 1000 Core Cluster at HPI Future SOC Lab. The microbenchmarks and applications selected to test performance were: *ping-pong*, *FFT* and *TeraSort*. The results show good and very good performance of the library on the cluster.

1 Introduction

The aim of this project was to check the scalability of parallel, network-intensive microbenchmarks and application written in Java, using the PCJ library, on the cluster with high-performance Ethernet – on the 1000 Core Cluster – Ethernet-based cluster – at the Hasso Plattner Institute.

The PCJ library [13] is HPC Challenge 2014 award-winning Java library for high-performance parallel computing. The PCJ library implements PGAS (Partitioned Global Address Space) paradigm for running concurrent applications on a multicore computer or computing clusters, so the systems that consist of many multicore nodes. The PCJ library is an open source library (BSD license) [12].

There are a number of microbenchmarks and applications that help to benchmark the PCJ library on different computing systems. The following microbenchmarks and applications were selected to test the performance of the PCJ library:

- ping-pong – measures maximum bandwidth of communication between two endpoints in regard to the message size,
- FFT – performs a Fast Fourier Transform algorithm and measure gained GFlops,
- TeraSort – measures the time needed to sort large datasets.

2 The PCJ Library

The PCJ library allows writing applications that utilize both multiple cores of a node and multiple nodes of a cluster. PCJ complies with Java standards and works with the newest Java version (currently Java 13). One of the key concepts was not to require any additional library, that is not part of the standard Java distribution, so

the user's dependencies are not affected. PCJ is distributed as a single jar file and can be downloaded from the library homepage [13] or from Maven Central Repository.

The `PCJ` class is the main class of the library. It contains a set of static methods that allow implementing necessary parallel constructs. The PCJ library provides methods for threads numbering, data transfer and threads synchronization. The communication is one-sided and asynchronous which makes programming easy and less error-prone. All executable units (tasks) in PCJ are called *PCJ threads*. All communication details are hidden from the programmer, irrespectively if PCJ threads are places on the same or other nodes.

More detailed information about the library can be found in other papers [7, 10].

The PCJ applications can run on a variety of hardware, from standard workstations, through x86 clusters and to supercomputers including Cray XC40 systems [8]. PCJ has been also tested on Intel KNL [6] processors.

The library has been already successfully used for parallelization of selected applications, like finding parameters of the neural network simulating connectome of C. Elegans [1, 2, 14], parallelization of the sequence alignment that reduces time needed for searching in genome database from 60 days to 1 day [4, 5], or for traversing of a large sparse graph that imitates real scale-free networks with small-diameter [16, 17], to mention only few.

3 Benchmarks

This section presents an overview of the selected benchmark and results obtained on the HPI Future SOC Lab 1000 Core Cluster.

3.1 Hardware and software

Hardware

The 1000 Core Cluster consists of 22 computational nodes. The computing nodes are equipped with the 4 Intel Xeon E7-4870 processors, with max. 2.40 GHz clock speed. Each processor contains 10 cores, each core can run 2 threads in hyper-threading mode. In total it is possible to run 80 threads per node. On each node, there is installed 1 TB of RAM, that more than 800 GB should be available for the user. Nodes are connected using 10-Gigabit ethernet using Intel 82599ES 10-Gigabit Ethernet Controller. The storage for the users (`/home` directory) was a NAS mounted a drive with 4 TB capacity.

In the benchmarks and applications, the maximum of 8 nodes for the computation was used.

Software

The operating system installed on the nodes was *Ubuntu* Linux in version 16.04.4 LTS (*Xenial Xerus*).

The *SLURM* (version 18.08.7) was used for submitting batch jobs to the cluster.

Codes written in C with MPI functions were compiled using *mpicc* command that uses *GCC* (version 5.4.0 20160609) as compiler and *MPICH* (version 3.2) as MPI implementation.

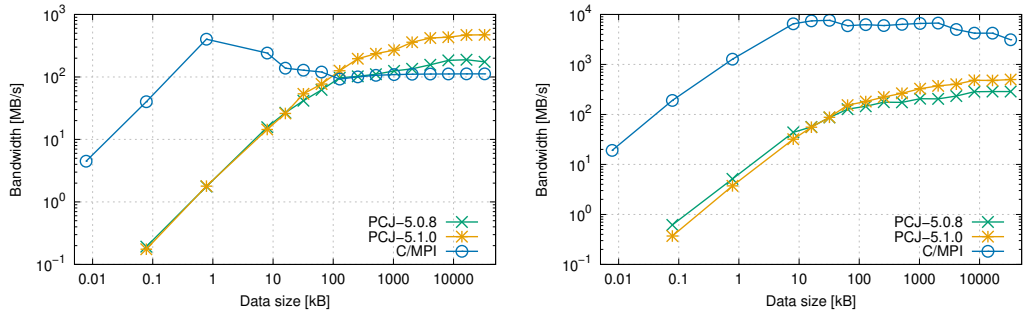
Java Virtual Machine from *Oracle JDK 13* package was used for the benchmarks of Java codes. For the *TeraSort* application *Apache Hadoop* (version 3.2.1) was used with additional required `javax.activation-api-1.2.0.jar` library placed into `${HADOOP_HOME}/share/hadoop/common/lib` directory.

3.2 Ping-pong

The ping-pong microbenchmark tests the bandwidth between a pair of threads. One thread sends a `double` array to another thread 100 times and waits for confirmation of successfully send. The total time needed for the operation is measured and the minimum average value from 5 repeats is taken as a result. In multiple tests, the size of array varies from 1 element (8 B) to 4 194 304 elements (32 MB).

Results

In this benchmark, two versions of the PCJ library was compared: 5.0.8 and 5.1.0 that is currently under development.



(**a**) Ping-pong using 2 nodes, one thread on each node

(**b**) Ping-pong using 1 node with 2 thread

Figure 1: Result of ping-pong microbenchmark

Figure 1a shows the result of ping-pong microbenchmark on 2 nodes. The performance of C/MPI is much better than for the PCJ implementation while sending small messages. However, for the larger data sizes, the PCJ implementation is better and was able to utilize almost 40 % of the potential network bandwidth. The performance of both versions of the library is similar, but the performance of the new version is 2-3 times better for larger messages.

Figure 1b shows the result of ping-pong microbenchmark while using 2 threads inside 1 node. The PCJ library implementation of communication, that tries to make a full copy of the transferred object, is worse in that scenario compare to C/MPI

implementation irrespectively to message size. The performance of both versions of the library is similar, but the performance of new version is slightly better for larger messages.

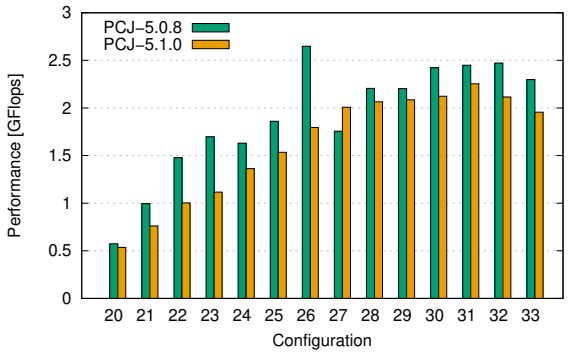
3.3 FFT

FFT (Fast Fourier Transform) is an algorithm that computes the discrete Fourier transform (DFT). The implementation of the algorithm in the PCJ is based on the PGAS implementation developed for Coarray Fortran 2.0 [3].

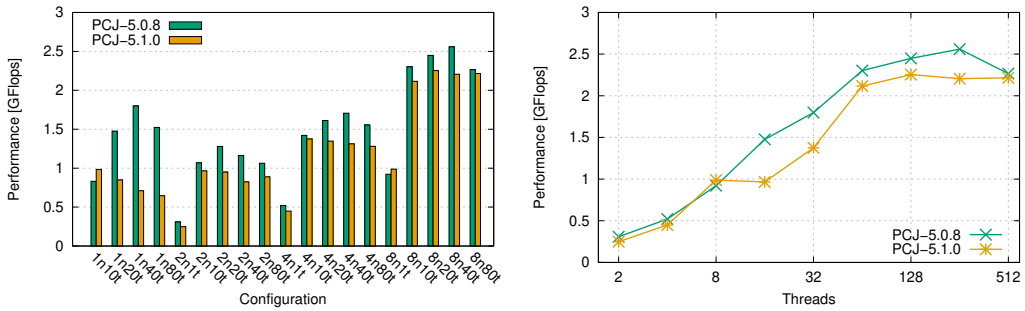
The used FFT algorithm requires to use a number of threads equal to a power of 2, so the cores of the cluster nodes were not totally used for computational purposes.

Results

In this benchmark, like in ping-pong microbenchmark, two versions of the PCJ library were compared: 5.0.8 and 5.1.0 that is currently under development.



(**a**) Performance results for FFT on an increasing number of elements to process (2^N) running on 8 nodes and 20 threads on each node



(**b**) Performance for FFT on 2^{30} elements. (**c**) Scalability for FFT on 2^{30} elements with $XnYt$ stands for X nodes and Y threads on an increasing number of threads each node

Figure 2: Performance results of FFT application

Figure 2a, 2b 2c show performance results of FFT applications. Figure 2a shows the obtained GFlops in regard to the size of the input data (from 2^{20} to 2^{33} elements) while using 8 nodes and 20 threads on each node. Figure 2b shows the obtained GFlops for the various nodes and threads configurations of the run for the input array containing 2^{30} elements. Figure 2c shows the maximum obtained GFlops for the same input array as before, but in respect to total threads used.

The performance for the previous version of the PCJ library is slightly better. It could be caused that new version is using more CPU power to serialize/deserialize data transferred and there could occur contention of threads.

3.4 TeraSort

Previous works show that the PCJ implementation scales well and outperforms Apache Hadoop implementation even by the factor of 100 [9, 15]. However, the applications presented in the aforementioned papers were rather simple: *WordCount*, *BFS* (Breadth-First Search), *quasi-Monte Carlo* (using 2-dimensional Halton sequence).

Currently, the *TeraSort* benchmark was used. The *TeraSort* benchmark requires creating proper input data using `teragen` application. The application generates a file(s) with multiple 100-bytes records that consist of 10-bytes key and 90-bytes value. The *TeraSort* sorts records using key values and outputs result in one file. The key is compared using a standard byte to byte comparison technique.

The Hadoop implementation of the *TeraSort* benchmark was the standard one provided with the Apache Hadoop cluster in the *examples* jar file.

The PCJ implementation is divided into 5 stages, similar to Edahiro's *Mapsort* described in [11]:

1. calculating pivots, the selected records from the input, that will be used for dividing input data into buckets; there is an option called *a number of pivots* that stands for the number of pivots per thread, i.e.:

$$\#\text{totalPivots} = \#\text{pivots} \cdot \#\text{threads}$$

2. reading input and placing records into proper buckets:

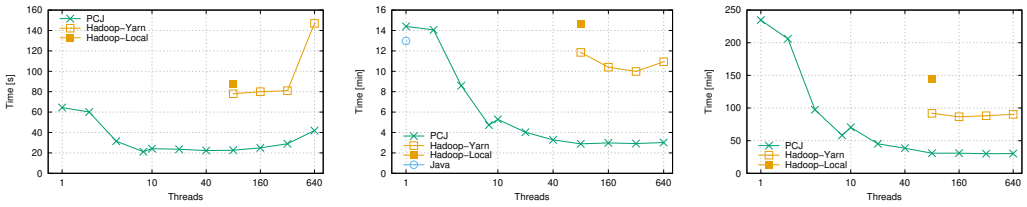
$$\#\text{buckets} = \#\text{totalPivots} + 1$$

3. exchanging buckets data between threads
4. sorting data stored in buckets
5. saving data from buckets to one single output file; each thread has to write to file in a correct order.

To compare PCJ with Apache Hadoop it was necessary to properly set up and launch Hadoop cluster on the SLURM submission system. For that purpose, the one node was selected as *Hadoop Master* that starts *namenode*, *datanode* and *resourcemanager* daemons.

For running applications on the Hadoop cluster, the additional nodes were used by starting *nodemanager* on each node just before submitting a job to the *resource manager*. The nodes use the *Hadoop Master* as a host for a default file system (*hdfs://*) and for resource manager.

Results



(**a**) 10^7 elements (**b**) 10^8 elements (**c**) 10^9 elements

Figure 3: Time needed to sort data depending on the allocated thread number

Figure 3a, 3b 3c present total time needed to sort and save 10^7, 10^8 and 10^9 elements respectively. The execution was done on the various configurations: using 1, 2, 4 and 8 nodes and 1, 10, 20, 40 and 80 threads per node. The shown time is the minimum time needed to sort in regard to total used number of threads. The *Hadoop-Local* stands for executing *Hadoop* job in standalone mode. The *Java* on figure 3b stands for executing single-threaded sorting code.

The PCJ implementation performance is about 3 times higher than the Hadoop implementation. It looks like the Hadoop does not utilize the whole available processing power while executing a job.

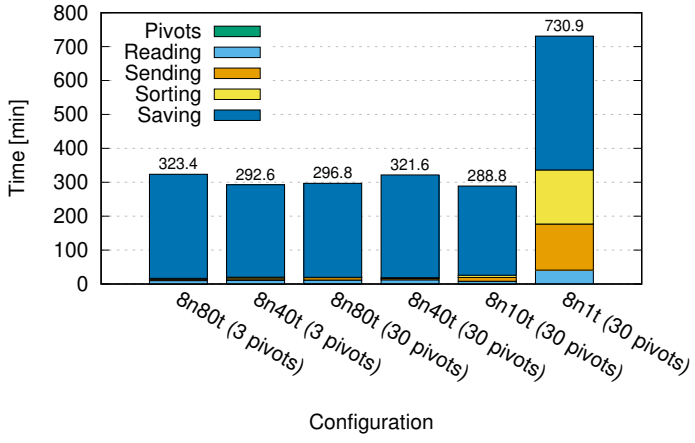


Figure 4: Time needed to sort 10^{10} 100-bytes records on various runtime configurations using PCJ. *XnYt* stands for X nodes and Y threads on each node

Table 1: Table with values for figure 4. *XnYt* stands for *X* nodes and *Y* threads on each node

Configuration	Execution time				
	Calc. pivots	Reading	Sending	Sorting	Saving
8n8t (3 pivots)	3.4 s	9.8 min	4.3 min	2.3 min	306.9 min
8n4t (3 pivots)	11.4 s	10.9 min	4.4 min	4.5 min	272.6 min
8n8t (30 pivots)	42.1 s	10.7 min	7.5 min	48 s	277.2 min
8n4t (30 pivots)	12.9 s	12.3 min	4.2 min	1.8 min	303.1 min
8n1t (30 pivots)	1 s	8 min	11.2 min	6.8 min	262.8 min
8n1t (30 pivots)	0.9 s	40.9 min	136.1 min	159.4 min	394.5 min

Figure 4 and table 1 present the performance results of PCJ *TeraSort* algorithm on 10^{10} elements (about 1 TB of the input file). Time was divided into parts needed to finish corresponding PCJ implementation stages. However, the values are approximated, as the stages could overlap. The most of the time is spent on saving data to an output file. There is also visible, that there is contention in reading the input file. Moreover, smaller bucket size, even when there is more of them, gives better sorting time.

The Hadoop implementation did not finish execution on 10^{10} elements due to lack of free disk space, as it claims in a log file. However, the disk storing HDFS data has more than 1.5 TB of free space at the time.

4 Conclusion

The results show good and very good performance of the PCJ library on the cluster. However, there are some weaker parts of the library like intranode communication, that requires more attention in the developments.

The little degradation of the performance for the *FFT* algorithm in the newest library version also could be caused by the intranode communication or contention of threads. Probably, the source code for FFT should be examined to optimize it for the new version of the library. Moreover, evaluation of the reference C/MPI implementation should be done on the cluster, to further analyse performance of the PCJ library.

The performance results of *TeraSort* application are very good, but there are also some places to improvements like overlapping saving and sorting data or receiving and sorting data.

The performance of Apache Hadoop prompts to an investigation of the better Hadoop configuration.

Acknowledgements

The author would like to thank Future SOC Lab, Hasso Plattner Institute for Digital Engineering for awarding access to the 1000 Core Cluster. The author would also like to thank colleagues Łukasz Górski, Łukasz Mikulski, and Jakub Narębski for their help during preparing and running benchmarks.

References

[1] Ł. Górski, P. Bała, and F. Rakowski. "A case study of software load balancing policies implemented with the PGAS programming model". In: *2016 International Conference on High Performance Computing Simulation (HPCS)*. July 2016, pages 443–448.

[2] Ł. Górski, F. Rakowski, and P. Bała. "Parallel differential evolution in the PGAS programming model implemented with PCJ Java library". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2015, pages 448–458.

[3] J. Mellor-Crummey, L. Adhianto, G. Jin, M. Krentel, K. Murthy, W. Scherer, and C. Yang. *Class II submission to the HPC Challenge award competition Coarray Fortran 2.0*. Submission to the 2011 HPC Challenge Class 2 Awards. 2011.

[4] M. Nowicki, D. Bzhalava, and P. Bała. "Massively Parallel Implementation of Sequence Alignment with Basic Local Alignment Search Tool Using Parallel Computing in Java Library". In: *Journal of Computational Biology* 25.8 (2018), pages 871–881.

[5] M. Nowicki, D. Bzhalava, and P. Bała. "Massively Parallel Sequence Alignment with BLAST Through Work Distribution Implemented Using PCJ Library". In: *International Conference on Algorithms and Architectures for Parallel Processing*. Springer. 2017, pages 503–512.

[6] M. Nowicki, Ł. Górski, and P. Bala. "Evaluation of the Parallel Performance of the Java and PCJ on the Intel KNL Based Systems". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2017, pages 288–297.

[7] M. Nowicki, Ł. Górski, and P. Bała. "PCJ–Java Library for Highly Scalable HPC and Big Data Processing". In: *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. 2018, pages 12–20.

[8] M. Nowicki, Ł. Górski, and P. Bała. "Performance evaluation of parallel computing and Big Data processing with Java and PCJ library". In: *Cray Users Group* (2018).

[9] M. Nowicki, M. Ryczkowska, Ł. Górski, and P. Bala. "Big Data Analytics in Java with PCJ Library: Performance Comparison with Hadoop". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2017, pages 318–327.

[10] M. Nowicki, M. Ryczkowska, Ł. Górski, M. Szynkiewicz, and P. Bała. "PCJ- a Java library for heterogenous parallel computing". In: *Recent Advances in Information Science* (*Recent Advances in Computer Engineering Series vol 36*), *WSEAS Press* (2016), pages 66–72.

[11] D. Pasetto and A. Akhriev. "A comparative study of parallel sort algorithms". In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. ACM. 2011, pages 203–204.

[12] *PCJ at GitHub*. Accessed: 2.09.2019. URL: https://github.com/hpdcj (last accessed 2020-04-01).

[13] *PCJ homepage*. Accessed: 21.03.2019. URL: http://pcj.icm.edu.pl (last accessed 2020-04-01).

[14] F. Rakowski and J. Karbowski. "Optimal synaptic signaling connectome for locomotory behavior in Caenorhabditis elegans: Design minimizing energy cost". In: *PLoS computational biology* 13.11 (2017), pages 1–28. DOI: 10.1371/journal.pcbi.1005834.

[15] M. Ryczkowska and M. Nowicki. "Performance Comparison of Graph BFS Implemented in MapReduce and PGAS Programming Models". In: *International Conference on Parallel Processing and Applied Mathematics*. Springer. 2017, pages 328–337.

[16] M. Ryczkowska, M. Nowicki, and P. Bala. "The performance evaluation of the Java implementation of Graph500". In: *Parallel Processing and Applied Mathematics*. Springer, 2016, pages 221–230.

[17] M. Ryczkowska, M. Nowicki, and P. Bała. "Level-synchronous BFS algorithm implemented in Java using PCJ library". In: *2016 International Conference on Computational Science and Computational Intelligence* (*CSCI*). IEEE. 2016, pages 596–601.

CitySensing: Digital terrain analysis and remote health monitoring

Dragan Stojanovic, Natalija Stojanovic, and Aleksandra Stojnev Ilic

Computer Science and Engineering Department
University of Nis, Faculty of Electronic Engineering
{firstname.lastname}@elfak.ni.ac.rs

The research and development within CitySensing project have been recently performed in two directions: digital terrain analysis and Big health and mobility data analytics. Watershed analysis, as a fundamental component of digital terrain analysis is based on Digital Elevation Model (DEM), and consists of computationally and data intensive computing algorithms that need to be implemented leveraging parallel and high performance computing methods and techniques. In this report, MFD (Multiple Flow Direction) algorithm for watershed analysis is implemented and evaluated on multi-core CPU and many-core GPU processing units providing significant gains in performance improvements and energy usage. The widespread use of mobile devices, such as smart phones, smart watches, bracelets, equipped with plenty of integrated sensors, provides collection of massive amounts of data and their processing and analysis on edge and cloud computing infrastructures. We developed and tested two Apache Spark applications for analysis of Big health and activity data streams in order to perform clustering and anomaly detection on streaming EKG data, as well as personal health and activities data.

1 Introduction

The CitySensing project is devoted to research and development of methods, techniques, software systems and platforms for Big Data applications that provide processing, analysis and mining of large-scale data collected by remote sensing, smart phones and medical Internet of Things devices. The research in CitySensing project have been conducted in two directions: digital terrain analysis, actually watershed analysis over large DEM datasets, and Big IoT and mobility data stream analysis over health and activities data streams.

For watershed analysis we implemented MFD (Multiple Flow Direction) algorithm and tested and evaluated it on multi-core CPU and many-core GPUs in order to achieve significant gains in performance improvements and energy usage. The implementation is based on NVIDIA CUDA implementation for GPU, as well as on OpenACC, parallel programming model and standard for parallel computing. Both phases of MFD algorithm i) iterative DEM pre-processing and ii) iterative MFD algorithm, are parallelized and run over contemporary (high-end) multi-core CPU and

NVIDIA GPU. The evaluation of proposed solutions is performed in respect to execution time, energy consumption and programming effort for program parallelization for different size of input data. Experimental evaluation shows advantage of using OpenACC programming over CUDA programming in watershed analysis implementation on GPU in terms of performance, energy consumption and programming effort, but also significant benefits in implementation for multi-core CPU.

The main focus of the work related to Big health and activity data analytics based on data collected from personal health devices (temperature, blood pressure, EKG, etc.), IoT devices with environmental sensors, as well as human activity recognition information generated by applying appropriate classification algorithms on data collected from smart devices (phone and watch). Such classification algorithms were implemented and reported during previous Future SOC Lab period. The basic machine learning algorithms used are binary and multi-class logistic regression, as well as k-means clustering and outlier detection.

2 High-Performance Parallel Implementation of Watershed Analysis

2.1 Background and related work

Among many interesting and computationally/data–intensive algorithms in GIS domain, digital terrain analysis, including viewshed and watershed analysis, have recently become the focus of parallel computing community. Different parallelization techniques and belonging architectures have used for accelerating watershed analysis [3]. Due to the recursive nature of the watershed algorithm, its parallelization is not a trivial task and have been the focus of research for its acceleration through execution on parallel architectures with distributed memory, shared memory, and many-core GPUs. Quin and Yhan [5] used CUDA to parallelize and accelerate both iterative DEM preprocessing step and a multiple-flow accumulation step of the watershed algorithm. Rueda et al. [6] implemented the flow accumulation step of the D8 algorithm in CPU, using OpenACC and two different CUDA versions, and compared the length and complexity of the programming code and its performance on different datasets. Eranen et al. [2] implemented all the steps of the drainage network extraction algorithm on a GPU, for single flow directions, considering the uncertainty in the input digital elevation model.

2.2 Parallel implementations of Watershed analysis algorithms

The focus of our research is on accelerating sequential watershed analysis using different parallel implementations, a native NVIDIA GPU implementation using CUDA, as well as OpenACC implementations mapped to both GPU and multicore CPU. We evaluate proposed solutions on HPI Future SOC Lab NVIDIA Docker infrastructure in respect to execution time, energy consumption and programming

effort for program parallelization for different size of input DEM data. Experimental evaluation proves expected improvements in performance of watershed analysis in respect to single-core CPU-based solution and shows feasibility of using GPU and multi-core CPU, using CUDA and OpenACC programming frameworks for digital terrain analysis and similar GIS algorithms. Furthermore, our evaluation shows better performance of OpenACC watershed analysis implementation in respect to CUDA one, with gains in lower energy consumption and less programming efforts. The main contributions of our work with greater details are published in [7]:

- We have developed parallel implementation of all phases and steps of watershed analysis using CUDA and OpenACC for NVIDIA GPUs and OpenACC for multi-core CPU.
- We have tested and evaluated OpenACC parallel implementation over several large DEM datasets both for multi-core CPUs and many/core GPUs, over commodity PC NVIDIA GPU, as well as high end NVIDIA Tesla K80 available through NVIDIA Docker plugin (nvidia-docker) over scientific cloud infrastructure.
- We show that the parallel implementation of OpenACC watershed analysis outperform corresponding CUDA implementation for different DEM sizes, while consuming less energy and requiring less programming efforts.

The source code and executables for Watershed analysis implementation, as well as experimental DEM datasets are available online at https://github.com/drstojanovic/WatershedAnalysis.

2.3 Evaluation of watershed analysis algorithms implementation

The proposed implementations have been tested on DEMs of different dimensions to evaluate efficiency and performance depending on the size of input data. Efficiency has been measured with respect to execution time, energy consumption, and programming efforts for algorithm parallelization. The experiment was carried out on the following parallel architectures.

- Intel(R) Core i5-4670K processor running at 3.40 GHz, 8 GB RAM (Random Access Memory), and NVIDIA GTX 770 graphic card.
- Tesla K80 GPU available on Hasso Plattner Institute (HPI) Future SOC (Service-Oriented Computing) Lab infrastructure. NVIDIA Tesla K80 Accelerator contains 4992 cores, 24 GB of GDDR5 memory, and 480 GB/s memory bandwidth, according to the specification. We implemented the Docker image for our Watershed algorithm implementation and ran a container on Tesla K80 using NVIDIA Docker plugin (nvidia-docker) and its options.
- Intel® Xeon® Processor E5-2620 v4, with eight physical and 16 logical cores, 128 GB RAM, which is also available at HPI Future SOC Lab infrastructure.

We have applied the Watershed analysis over DEMs of several dimensions: 1691 × 2877, 2414 × 2912, 2433 × 4152, 3278 × 4277, and 3308 × 5967 to evaluate the scalability of implementation with the data size increasing. These DEMs represent the models of digital terrain in Alaska displayed in figure 7 in the original form, using isohypses and hill shade relief, respectively, downloaded from the EarthExplorer USGS (United States Geological Survey) service.[1]

In order to measure the execution time and the energy consumption of Watershed analysis, the MeterPU library is provided for advanced architectures, such as NVIDIA Tesla K80 and Intel® Xeon® Processor E5-2620 v4 CPU that we used at a Future SOC Lab. The experimental results of the sequential and parallel CUDA and OpenACC implementations on these architectures are presented in tables 1–4 in figure 1. The Speedup achieved for different parallel implementations with respect to the sequential ones is presented in table 5 and in figure 2.

Based on tables 1–4 on Fugure 1 and figure 2, it can be concluded that OpenACC multi-core CPU implementation (marked as MC in figure 2) shows speedup on average 16.6 (maximum 18.5), CUDA many-core GPU implementation (marked as CUDA in figure 2) shows speedup on average 18.7 (maximum 20), and OpenACC many-core GPU implementation shows speedup on average 33.7 (maximum 40.6). Regarding energy consumption, all parallel solutions consume less total energy (CPU+GPU) than the corresponding sequential solution. The total energy consumption of multi-core OpenACC solution is comparable to the CUDA GPU solution. The first one consumes more CPU energy while the second one consumes more GPU energy. When comparing GPU-based solutions, OpenACC implementation consumes almost two times less energy than its CUDA counterpart.

Adaptation of sequential watershed algorithm implementation to many-core GPU requires significant code transformations and optimizations for CUDA parallel implementation, and that is why we considered OpenACC for parallel implementation for many-core GPU, but also for multi-core CPU. OpenACC requires less development effort, lower risk of errors, and better code readability with respect to the CUDA GPU solution. We have implemented the watershed analysis algorithm that consists of two computationally intensive and time-consuming phases: (1) iterative DEM preprocessing and (2) iterative MFD algorithm, which are both suitable for parallelization. Experimental evaluation indicates improvement in performance with respect to a single-core CPU-based solution and shows feasibility of using GPU and multicore CPU for watershed analysis. The evaluation of proposed solutions is performed with respect to execution time, energy consumption, and programming effort for program parallelization for a different size of input data. The experimental results show benefits of using the OpenACC framework over CUDA for parallelization of watershed analysis using the MFD algorithm and, thus, its feasibility for GIS analytic algorithms over Big raster and vector geospatial data.

[1] http://earthexplorer.usgs.gov (last accessed 2020-04-01).

Single core—Intel® Xeon® Processor E5-2620 v4			
DEM Size	Execution Time (s)	Energy Consumption—CPU (J)	Energy Consumption—GPU (J)
1691 × 2827	159.776	12135.8	4871.3
2414 × 2912	315.515	24538.2	9619.4
2433 × 4152	459.171	35769.8	14,007.4
3278 × 4227	781.827	61060.6	23,862.2
3308 × 5967	1090.010	85613.0	33,238.4

Table 2. Multi-core—OpenACC-Intel® Xeon® Processor E5-2620 v4.

	Multi-core—OpenACC-Intel® Xeon® Processor E5-2620 v4		
DEM Size	Execution Time (s)	Energy Consumption—CPU (J)	Energy Consumption—GPU (J)
1691 × 2827	9.5034	1307.65	290.28
2414 × 2912	17.0548	2469.39	519.83
2433 × 4152	37.1679	5228.46	1132.68
3278 × 4227	43.5649	6579.67	1328.08
3308 × 5967	61.8061	9400.93	1883.37

Table 3. OpenACC-Tesla K80.

	OpenACC-Tesla K80		
DEM Size	Execution Time (s)	Energy Consumption—CPU (J)	Energy Consumption—GPU (J)
1691 × 2827	6.2446	435.93	690.64
2414 × 2912	10.0951	707.68	1147.29
2433 × 4152	14.2851	942.79	1697.09
3278 × 4227	19.8665	1469.80	2442.86
3308 × 5967	26.8563	1973.31	3359.60

Table 4. CUDA—native-Tesla K80.

	CUDA—Native-Tesla K80		
DEM Size	Execution Time (s)	Energy Consumption—CPU (J)	Energy Consumption—GPU(J)
1691 × 2827	9.7792	714.52	1099.04
2414 × 2912	16.6563	1196.26	1945.94
2433 × 4152	24.0191	1737.13	2849.46
3278 × 4227	39.0244	2875.51	4789.63
3308 × 5967	55.9863	4104.54	6941.10

Figure 1: The execution time and energy consumption of Watershed analysis implementation on different platforms

Table 5. Speedup (Tesla K80).

Speedup (Tesla K80)			
DEM	**SC/MC**	**SC/CUDA**	**SC/OpenACC**
1691 × 2827	16.8125	16.3383	25.5863
2414 × 2912	18.5001	18.9427	31.2543
2433 × 4152	12.3540	19.1169	32.1434
3278 × 4227	17.9463	20.0343	39.3540
3308 × 5967	17.6360	19.4692	40.5868

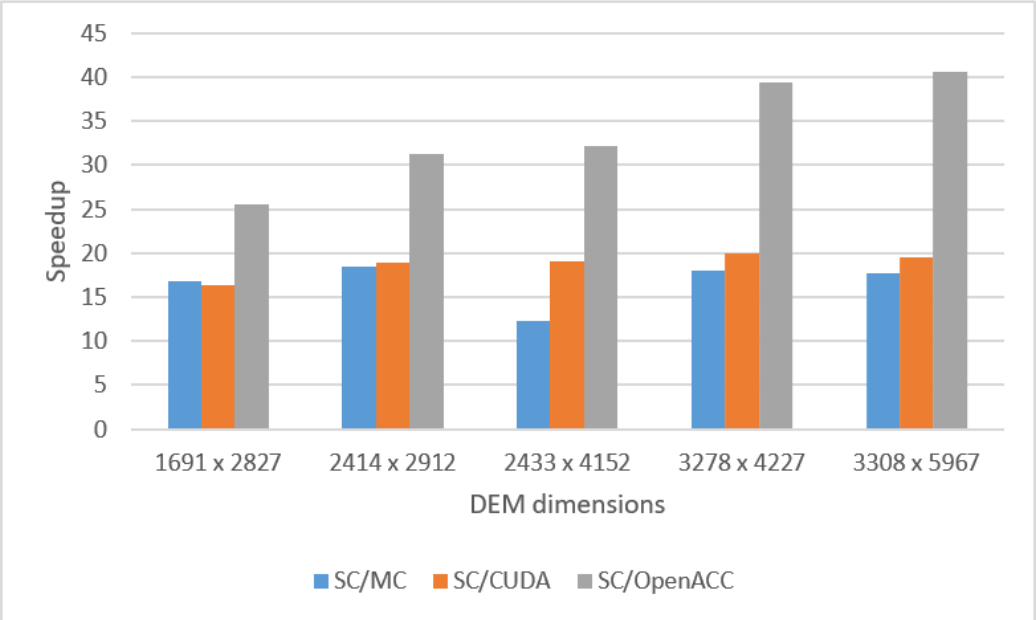


Figure 2: The Speedup for multi-core Intel® Xeon® E5-2620 v4 and many-core Tesla K80 implementations

3 Big health and activity data streams clustering and outlier detection

3.1 Background and related work

The fusion, analytics and mining of mobile sensor data, collected from personal mobile and health devices, as well as Smart city infrastructure can provide a personalized health and monitoring system for general well-being where individuals can be provided with healthcare tailored to their needs [1]. The storage, aggregation, processing and analysis of Big health data could be performed within public, private or hybrid cloud infrastructure [4]. The results of data analysis and mining are provided to physicians through tailored dashboard applications.

In this research we continued working on Big mobility and IoT data analytics performed and reported in the previous HPI Future SOC Lab period (Spring 2019). High-level mobility and sensor data generated at users' mobile and wearable devices are fused, aggregated, processed and analyzed at the RemoteHealth server running on a cluster/cloud infrastructure (HPI Future SOC Lab cluster). For processing and analysis of Big mobility and IoT data, RemoteHealth server is based on distributed processing frameworks, such as MapReduce/Hadoop and Apache Spark.[2] For efficient processing and analysis of masssive mobility and IoT data are stored in a distributed file system in the cloud/cluster (HDFS). We implemented Human Activity Recognition (HAR) application within RemoteHealth server, based on Apache Spark platform and Spark MLlib machine learning library. The application processed and analysed publicly available ExtraSensory data[3] generated by mobile health application for data collection provided by Extrasensory project[8]. ExtraSensory dataset contains data from 60 users and more than 300K labeled examples (minutes) originated from smartphone and smartwatch sensors[9].

In previously reported research we performed sensor data fusion from six sensors: smartphone accelerometer, gyroscope, watch accelerometer, location, audio, and phone state, and data analysis using Spark MLlib library[4] to detect user activities like standing, sitting, laying down, walking, running, ridding bicycle, etc.

3.2 IoT, health and activity data clustering

We extended RemoteHealth server by implementation two health monitoring and emergency (HME) applications on this platform, using Apache Spark Streaming and Spark MLlib algorithms for clustering and outlier detection. For communication between different RemoteHealth components in processing and analysis of data streams, Apache Kafka,[5] a distributed message platform is used.

[2] https://spark.apache.org (last accessed 2020-04-01).
[3] https://extrasensory.ucsd.edu (last accessed 2020-04-01).
[4] https://spark.apache.org/mllib/ (last accessed 2020-04-01).
[5] https://kafka.apache.org (last accessed 2020-04-01).

The first application takes a stream of sensor data originated from smart phone/watch sensors, wearable health devices (heart rate, blood pressure, body temperature, respiration rate) and Arduino box with two sensors attached for air temperature and humidity. The smart phone/watch sensors data were fused and aggregated using Human Activity Recognition (HAR) application presented in the previous report. The HAR application continuously detects user physical activity and sends it to the specific Kafka topic to which the Health monitoring emergency application is subscribed in order to detect outliers and abnormalities in user physical status. During the preparation (offline) phase, using clustering over large training data, the HME application produces a model of user status represented by 8-16 clusters, using k-means clustering algorithm. In the online phase, HME application subscribe to Kafka topics to receive streaming sensor data and detects outlier and their distance from generated clusters to determine the emergency status. In case of significant outlier, the emergency notification is send to the mobile and Web dashboard application used by a physician.

The second application performs streaming clustering and anomaly/outlier detection over stream of EKG signal data obtained form the publicly available from Kaggle.[6] The anomaly detection is based on k-means model produced by offline data clustering, and also generate the emergency alert if received EKG signal represents an anomaly, actually if it is out of determined clusters larger than the threshold value.

3.3 Evaluation of health monitoring applicaitons

For the purpose of deploying, testing and evaluation of RemoteHealth Big data streaming applications we use HPI Future SOC Lab infrastructure, with cluster of 9 virtual machines: 1 master (XL VM), 8 slaves (L VM) installed and configured with Apache Hadoop, Spark and Kafka. The first application was developed in Java and the second one in Python.

The timeline of spark jobs and executors executed on cluster nodes for the first HME application is shown in figure 3.

The second HME application was developed in Python and prints the results on the console windows and visually plotted every second (figure 4).

4 Conclusion

The work presented in this report shows that using OpenACC and CUDA parallel programming paradigms can significantly improve the performance in executing various computation and data intensive GIS algorithms and that using parallelization and high-performance computing in GIS represents a promising direction for research and development. The benefits of parallel processing of geospatial data is

[6] https://www.kaggle.com/shayanfazeli/heartbeat (last accessed 2020-04-01).

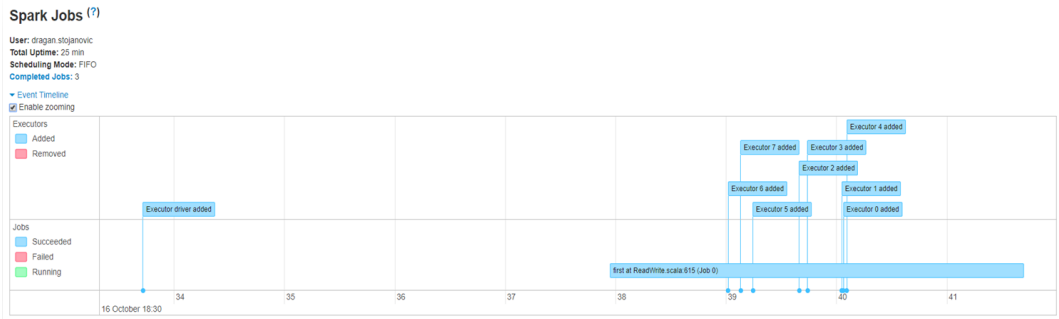


Figure 3: The Spark Executor and Workers perform the HME1 application implementation on different platforms

```
-----------------------------------------
Time: 2019-10-04 00:01:39
-----------------------------------------
Euclidean distance between closest cluster center and signal: 1.9465560795006027
Heartbeat is in normal range
Signal saved to: ./plots/normal/00-01-59

-----------------------------------------
Time: 2019-10-04 00:01:40
-----------------------------------------
Euclidean distance between closest cluster center and signal: 1.8375229100339647
Heartbeat is in normal range
Signal saved to: ./plots/normal/00-02-05

-----------------------------------------
Time: 2019-10-04 00:01:41
-----------------------------------------
Euclidean distance between closest cluster center and signal: 2.337934232039864
ANOMALY DETECTED!
Signal saved to: ./plots/anomalies/00-02-09
```

Figure 4: HME2 application: Anomaly detection in streaming EKG data

confirmed in parallelization of watershed analysis algorithms and they are evaluated on multi-core CPU and many-core GPU using CUDA and OpenACC frameworks.

We expect that availability of HPI Future SOC Lab infrastructure for our project will further improve our research expertise and experience in domains of mobile sensing, IoT and Big mobility, activity and health-related data processing, analysis and mining and will result in scientific achievements through preparation/publication of technical reports, scientific papers and development of prototype/demo solutions.

The availability of HPI Future SOC Lab infrastructure gives us useful insights in deploying and execution of Big data applications on the real cloud infrastructure.

References

[1] S. B. Baker, W. Xiang, and I. Atkinson. "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities". In: *IEEE Access* 5 (2017), pages 26521–26544. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2775180.

[2] D. Eranen, J. Oksanen, J. Westerholm, and T. Sarjakoski. "A full graphics processing unit implementation of uncertainty-aware drainage basin delineation". In: *Computers and Geosciences* 73 (2014), pages 48–60. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2014.08.012.

[3] Y. Hu. "Parallelization research on watershed algorithm". In: *IET Conference Proceedings* (Jan. 2012), pages 1524–1527. DOI: 10.1049/cp.2012.1272.

[4] S. U. Khan, A. Y. Zomaya, and A. Abbas, editors. *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-58280-1. DOI: 10.1007/978-3-319-58280-1.

[5] C.-Z. Qin and L. Zhan. "Parallelizing Flow-accumulation Calculations on Graphics Processing units-From Iterative DEM Preprocessing Algorithm to Recursive Multiple-flow-direction Algorithm". In: *Computers and Geosciences* 43 (June 2012), pages 7–16. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2012.02.022.

[6] A. J. Rueda, J. M. Noguera, and A. Luque. "A comparison of native GPU computing versus OpenACC for implementing flow-routing algorithms in hydrological applications". In: *Computers and Geosciences* 87 (2016), pages 91–100. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2015.12.004.

[7] N. Stojanovic and D. Stojanovic. "Parallelizing Multiple Flow Accumulation Algorithm using CUDA and OpenACC". In: *ISPRS International Journal of Geo-Information* 8.9 (Sept. 2019), page 386. ISSN: 2220-9964. DOI: 10.3390/ijgi8090386.

[8] Y. Vaizman, K. Ellis, and G. Lanckriet. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches". In: *IEEE Pervasive Computing* 16.4 (Oct. 2017), pages 62–74. ISSN: 1536-1268. DOI: 10.1109/MPRV.2017.3971131.

[9] Y. Vaizman, K. Ellis, G. R. G. Lanckriet, and N. Weibel. "ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. 2018, page 554. DOI: 10.1145/3173574.3174128.

Towards production-ready tools for self-driving data management with deep reinforcement learning

Project Report

Gabriel Campero Durand, Pavlo Shevchenko, Milena Malysheva, Ivan Prymak, and Gunter Saake

Databases and Software Engineering Workgroup
University of Magdeburg
campero@ovgu.de, saake@iti.cs.uni-magdeburg.de

In this short report we summarize the progress and results of our 6 month project using the Future SOC Lab Compute resources to support our research on *"Production-Ready Data Management with Deep Reinforcement Learning (DRL)."* With this project we established a focused research agenda to study what we identified, at the moment, to be some core challenges for practitioners aiming to use DRL techniques in data management. At this stage of the project our core contributions are: a) a tool for integrating DRL for any given application with an early comparative evaluation, b) results of vertical and horizontal partitioning with DRL, compared to state-of-the art solutions, c) reproducibility of produced results on safety challenge environments and early evaluations of proposed solutions for the challenges. As next steps we will consider evaluations of offline-online life-cycle designs, studies and solutions for understanding models, extending our integration tool with further frameworks, and a conclusion of our work on data partitioning with DRL.

1 Introduction

Data management tools are routinely tasked with automatically solving complex optimization problems (e.g. distributing tasks across compute devices), which would be time-consuming to solve to their optima, and for which real-world performance impact factors are difficult to model accurately (as noted by other researchers [10, 14]). Commonly, heuristics are employed to these kind of problems, but these can be brittle, unable to guarantee optimal behavior, and they can be hard to evolve or maintain.

With the development and standardization of machine learning (ML) applications, there's a great interest for creatively adopting some ML techniques to enhance data management, either in substituting hard-coded heuristics by components that learn from experience, or by creating novel features using learning. Either way, engineering teams considering such adoption nowadays face many challenges, creating reticence towards this change, as including machine learning into existing software systems increases product and system complexity, introduces requirements for continuous

testing practices and creates uncertainty for quality assurance. Given this context, we take the perspective that it is of foremost importance to study ML applications in data management with a clear focus on production-readiness, if these technologies are to live up to their potential. In line with this goal we proposed and carried-out a 6 months project covering several of our concerns for production-readiness, utilizing the compute resources made available by the Future SOC Lab.

During this project we specifically focused on deep reinforcement learning (DRL) applications in data management. These are a kind of learning model based on reinforcement learning (RL), where neural networks are used for function approximation, helping the agent to generalize to unseen cases while keeping a bounded memory footprint [4]. In recent years DRL has been a part of many breakthroughs in AI, with solutions showing super-human performance at Atari games [11], Go [13], and others. Stemming from such interest, the application of these models is also being widely studied for many computer systems tasks, including network congestion control, resource allocation in the cloud, device placement, packet classification, SQL join order optimization, among others [6, 10].

For the adoption of DRL in data management, we specifically considered, at the beginning of this project, that practitioners faced 6 main challenges: 1) the need for improvements in the hyper-parameter tuning process for new use cases, 2) the requirement for guidelines in framework selection, 3) life-cycle designs to exploit offline and online training, 4) the adoption of models for scaling to large action and observation spaces, 5) evaluations for model safety and 6) support for deep learning model understanding. To address these challenges we proposed a decomposition of our work into 5 sub-projects:

1. Comparison of DRL frameworks and life-cycle design.
2. New data management use cases: partitioning with DRL, and hyper-parameter tuning.
3. Adoption of models for large discrete action spaces.
4. A study towards safety guarantees in data management DRL applications.
5. A lightweight tool for visualizing DRL latent representations, based on training logs.

2 Contributions

At this stage of our project we can enumerate 3 core contributions, in line with the sub-projects defined. In this section we present them and we also discuss briefly about ongoing progress in some other goals of the project.

The contributions we discuss here overlap with work currently published [3], or to be published, including the Master Thesis of Pavlo Shevchenko,[1] Milena Malyheva and Ivan Prymak,[2] and Shikha Mehta,[3] which are all done in our research group.

2.1 A Tool for Cross-DRL Framework Research

Within this sub-project we developed a framework to support a straightforward integration of environments to different DRL tools.[4] The core principle of this design is to create a similar approach to handle any framework, while extending (rather than modifying) the code-base of the framework. Figure 1 shows the core components of the solution. For each framework there is an *Experiment Runner* component that encapsulates the logic of running the experiment and managing the results, next are some *Components*, which extend each framework, as required. For example, for Ray it was necessary to create, through inheritance, a parametric DQN agent in order to perform action pruning in a Rainbow model. In addition the framework *Components* help in mapping between the provided commonly-formatted *Configuration* files that we propose, to the internal configurations given by the different frameworks (which do not sync entirely).

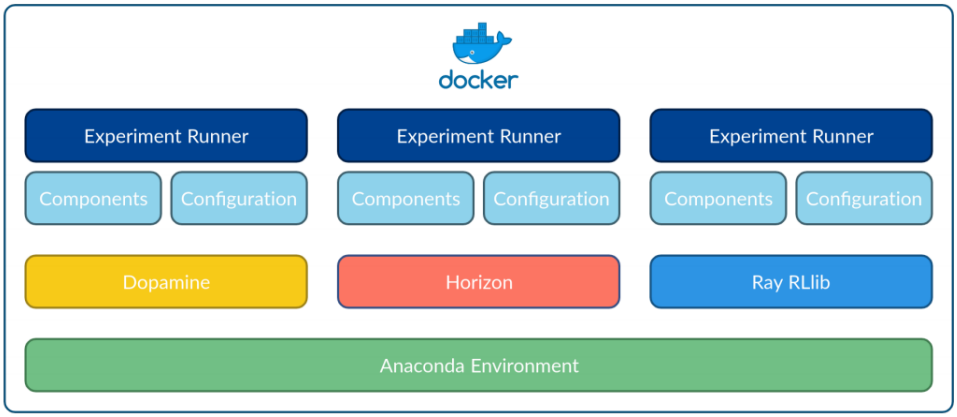


Figure 1: Our tool for evaluating DRL frameworks, allowing researchers to mix-and-match across frameworks, given them extra models for their evaluations

[1] Pavlo Shevchenko. A Comparative Evaluation of Deep Reinforcement Learning Frameworks. Master's thesis, University of Magdeburg, September 2019. http://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/shevchenko2019comparing.pdf

[2] Milena Malysheva and Ivan Prymak. Evaluation of Unsupervised and Reinforcement Learning approaches for Horizontal Fragmentation. Master's thesis, University of Magdeburg, October 2019. To be published.

[3] Shikha Mehta. Towards Safety Unit Tests for Deep Reinforcement Learning in Computer Systems. Master's thesis, University of Magdeburg, December 2019. To be published.

[4] Available in: https://github.com/pshevche/drl-frameworks

In our study we specifically considered 3 frameworks:

- Google Dopamine [1], a compact framework to help researchers quickly prototype DRL solutions, offering implementations for state-of-the-art DQN agents, easy integration with Tensorflow solutions and a clear formalization of hyperparameters in the form of gin files.
- Ray RLLib [9], a framework developed by the BAIR Lab at Berkeley University. This represents a comprehensive extensible framework with a focus on efficient large-scale processing, including high-throughput architectures for agents. It is based on tasks and actors, and includes a distributed object store. Ray RLLib includes offline-learning agents, such as MARWIL, and multi-agent solutions such as MADDPG. It supports both Tensorflow and Pytorch, as libraries for the definition of deep learning models.
- Facebook Horizon [5], a framework that focuses on production-readiness, with emphasis on feature engineering, offline-learning and compatibility across neural network formats. It is based on Pytorch. Unfortunately, at the time of our study Horizon did not support convolutional layers, limiting our consideration of this framework for studies on input data representation.

We considered these frameworks for two specific learning tasks. First, the simple cartpole task,[5] which is a traditional reinforcement learning application, consisting of an agent having to learn how to balance a pole by moving a cart horizontally. We also considered a more complex task of learning the proper join order for the join order benchmark queries from the IMBD dataset. To this end we used the implementation currently developed by the Park project[6][7] This is evaluated using a PostgreSQL database, and the Apache Calcite framework.

Overall we find that although agents from different frameworks are indeed susceptible to misconfigurations that would hamper the performance they reach, we did not observe this to occur commonly, and in general basic DQN agents perform well for the tasks that we studied. Our results further suggest that Ray RLLib's implementations lead to a better overall runtime, as illustrated in figure 2. This faster runtime is achieved by its acceleration of training, which ends up amounting to less than 25 % of its running time, on some cases, as shown in figure 3.

In our experiments we also find that checkpointing has little influence on the overall running time, and that the use of GPUs leads to up to 69 % performance improvements for the Pytorch-based Horizon, but this still does not make the model perform better than the counterpart DQN from other frameworks. Finally, we find that the most competitive agents are actually not common across frameworks, thus justifying the need for a cross-framework functionality to facilitate their study. In this regard, Dopamine's Implicit Quantile agents, and Ray RLLib PPO represented the

[5] https://gym.openai.com/envs/CartPole-v0/ (last accessed 2020-04-01).
[6] https://github.com/parimarjan/query-optimizer (last accessed 2020-04-01).
[7] https://github.com/park-project/park (last accessed 2020-04-01).

best of the simple agents offered by the frameworks, with both solutions not being available in other of the studied frameworks. Empirically we also observed that Ray is the best documented of the tools, and that Dopamine is better for quick prototyping, since the dependency on hyper-parameters are well segregated to configuration files, and the models are all implemented with relative little code complexity.

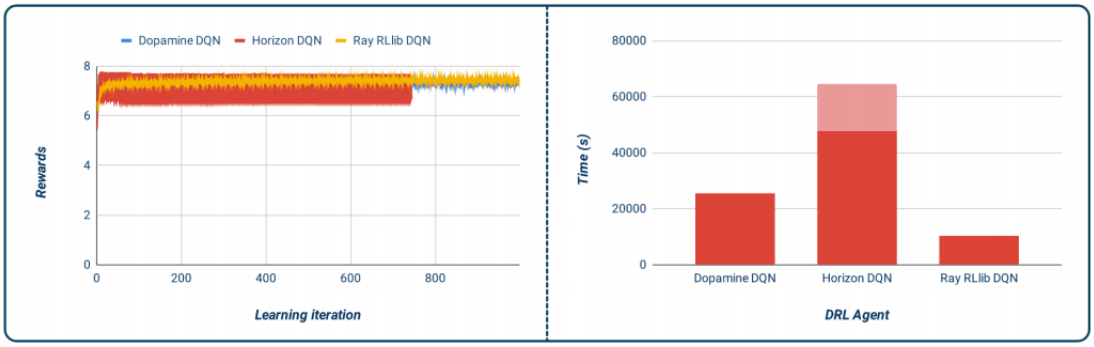


Figure 2: Performance of different DQN agents on learning how to optimize join orders

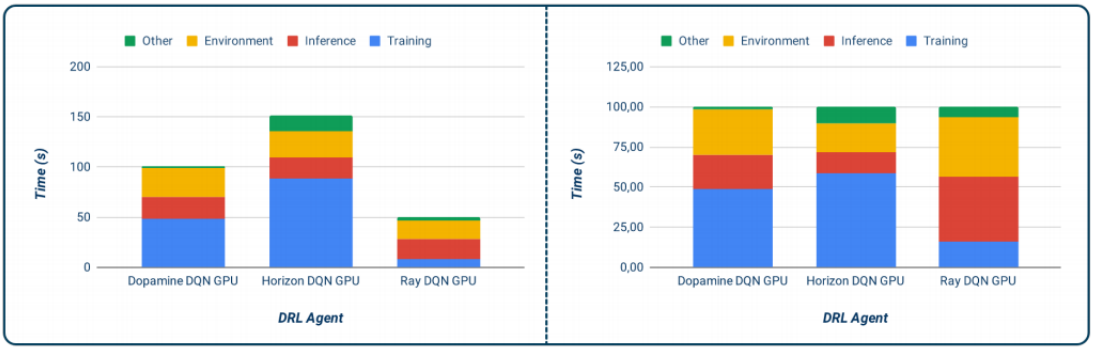


Figure 3: Breakdown of the time taken by the different agents to interact with the join order optimization environment

2.2 Partitioning with DRL

During the length of our project we have also extended our research of data partitioning with DRL. This research is motivated on the insight that choices about partitioning and data layouting are at the core of the design of data management tools. However, algorithms to either find the optimal partitioning, or to improve an

existing partitioning, given a workload, are commonly bounded by their choices for pruning the space of possible solutions. In such scenario experience-based learning can help to learn the actual optima, exploring more efficiently, and also lead to a much faster inference process. We envision that mature DRL solutions for partitioning will be able to serve as a building block for innovative and highly flexible storage engines.

To this end we have studied vertical partitioning with DRL using cost models as a delayed reward. We studied both bottom-up and top-down approaches, finding the former to be more suited to the task. We took a fully observable state containing the current partitioning and the target workload. In our evaluations we have used the TPC-H benchmark data at different scale factors, and state-of-the-art vertical partitioning algorithms as baselines. Our work during this project on these models encompassed the impact of design choices. Based on these experiments we were able to segregate design aspects into little-impact (e.g. Boltzmann vs. ϵ-greedy) or high-impact groups (e.g. changes in the observation space, or number of steps required to solve the learning task). Figure 4 shows some of our results considering other aspect of the process—how competitive the agent results in terms of runtime, when compared with state-of-the-art approaches. Our results show that DRL methods are competitive as the number of attributes to be partitioned from the table (LineItem is larger) increases.

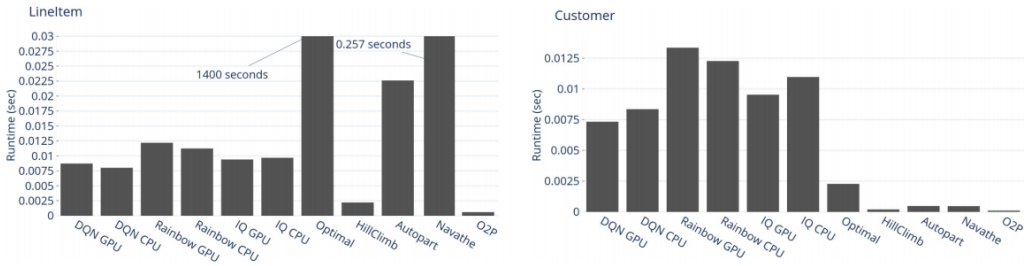


Figure 4: Comparison of the learned DRL agents against state-of-the-art methods in suggesting an optimal partitioning. All solutions reach the same cost.

Moving on to horizontal partitioning, the core challenge is that of the large number of possible partitions for any normal table. To overcome this in a learned model, different data representations need to be studied, for example affinity-based representations of the problem (where predicates, rather than the participating tuples are the unit of representation). In our research thus far we have developed based on such representations two learned solutions for horizontal partitioning, comprising unsupervised learning (hierarchical clustering) and DRL, with the former serving as a baseline to understand the latter. Currently, our evaluation with Dopamine agents, using a generated workload over the TPC-H data, show that learning is possible and efficient, without a large impact of hyper-parameters. Still, our results also show,

that when using a cost model the DRL-provided solutions fall behind the clustering algorithm, but are still better than other state-of-the-art solutions, as shown in figure 5. Though we find that training is not entirely possible with a database, we will evaluate in next steps the reality gap between the cost models and the actual end systems. Currently, we are extending the models to work on SparkSQL, leading to hierarchical partitioning. We do so with an additional baseline consisting of Ward's method for hierarchical clustering.

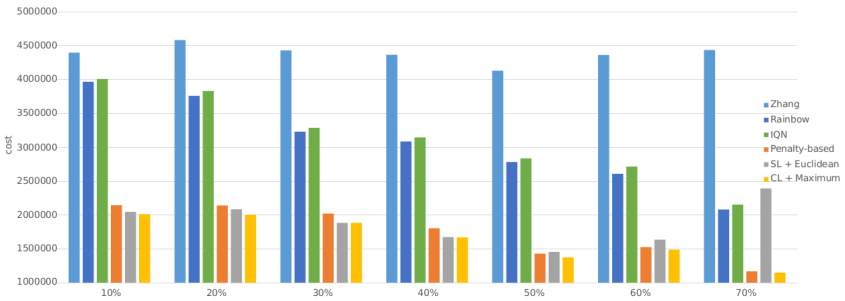


Figure 5: Costs (tuples fetched) achieved in horizontal fragmentation, by different learning models, as the number of duplicates in the workload varies

2.3 Safety Challenges of DRL

Specification (whether our instructions for the model are clear-enough, articulating exactly what we expect the model to value) and robustness (whether the model is sufficiently sound to withstand perturbations) are the two core groups of safety challenges for DRL agents. Leike et al. [8] proposed that to understand how well an agent would fare with regards to these challenges, sanity checks could be done, by evaluating how a model performs on a set of simple environments that capture the bare-minimum about these challenges. Authors in fact created and offered such environments, calling them AI Safety Gridworlds. In addition authors reported initial results for DQN and A2C agents with these environments, showing shortcomings and suggesting possible improvements.

Since authors did not provide open-source implementations, our first task to use concepts similar to those in their environments was to actually reproduce their results with our own agent implementations. To this end we used Ray RLLib, first to successfully recreate the results based on our own implementations, and second to study improvements proposed. Among the latter we have evaluated the use of several techniques. For example in the case of the distributional shift *lava world environment* (where the agent must learn a task and afterwards is evaluated on a slightly different task), we have considered the use of parametric noise, entropy regularized models and Bayesian Deep-Q Networks. Unfortunately, we have not found yet naive

solutions that perform particularly well on this generalization challenge, after an extensive training to overfit to a task. Figure 6 gives an example of such behavior. Here after training a BDQN agent (in its empirically-found best configuration) for 200 k episodes the agent has learned to master the task, reaching the optimal reward. In the following steps, until approximately the 400 k steps we stop the training of the agent and we confirm its predictable behavior on the task that it has mastered. Afterwards we introduce the distributional shift challenge (while not letting the agent learn from it), we find that the agent is not able to reach high rewards, as the overfitting makes the agent incurr in many mistakes. Currently we evaluate a mapping of these safety challenges to the join order optimization task (hence, creating a safety unit test for such kind of agents), and the impact of further specialized techniques such as introducing risk-aversion to DRL agents.

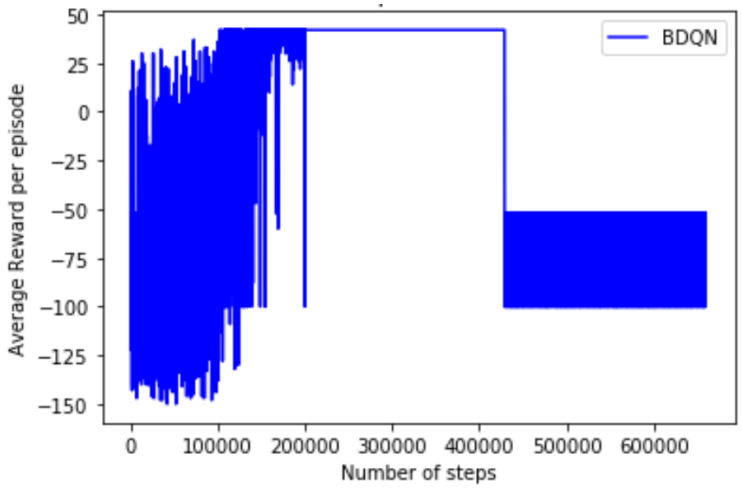


Figure 6: BDQN agent training on a distributional shift challenge

2.4 Progress in other Goals

As part of this research project we are currently also experimenting with offline learning for a standardized index selection environment, by using the MARWIL agent offered in Ray RLLib, with logs generated by a standard off-the-shelf index advisory tool for PostgreSQL. As part of this research project we also proposed the evaluation of practical adoptions of models that support large action spaces (Wolpertinger), however this study has been postponed till we conclude our overall partitioning solution. Furthermore, we have also considered the essential aspect of adopting a model understanding tool (i.e., interpretable ML [12]), that could provide more insights in the hyper-parameter tuning and model adoption process. Integrating such solution with our cross-framework tool, is currently in relatively early progress.

3 Used Resources

During our project we used a dedicated server provided to us with a GPU (gpu-01), in addition to the access to the GPU playground, which we used on occasions. The core cross-framework evaluations were supported entirely by the resources of the lab. We consider that some small issues reported to us, regarding connecting to the lab, or some work overheads from having to use Docker, might have affected our ability to actually leverage the resources offered to our team of researchers. If our project is renewed, in a subsequent version we will encourage our team of researchers to smartly use the provided resources even more.

4 Conclusions and Next Steps

The development of our solution to mix-and-match across DRL frameworks constitutes a valuable research tool for us, helping us to speed-up our prototyping with agents and ideas not immediately available if we were working with a single framework. As next steps in this line of research and development, we are interested in integrating another framework with growing model offerings (Intel Nervana Coach), and some non-framework implementations from established researchers of innovative models.[8] We expect that these steps will enhance our abilities to interface quickly across models for a single task. In the coming months we expect to improve our current solution with the addition of a model management data store (e.g. MLFlow), and a model understanding tool. Finally, we will produce a technical report about our solution for interfacing across frameworks, with walkthroughs for its use.

A comprehensive benchmarking study, including Arcade environment tasks with DRL frameworks is facilitated by our work, but it is outside the core focus of our current research initiative.

We aim to continue using the ability to quickly integrate models from different frameworks for studying two core data management tasks: partitioning and join order optimization with cardinality estimation. To this end we will require extensive hyper-parameter optimization and the corresponding compute-intensive evaluations. In the next steps we focus on representation learning for the inputs of the the join ordering agents (as extensions to the work of Kipf et al [7], spanning agents that are provided with different embeddings for the input queries, with samples of the table data or data profiles), and in consolidating our partitioning research in a consistent end-to-end, well-studied tool paired with a set of standardized environments to test models at the task.

Based on our current progress we have been able to revise the challenges facing engineers interested in adopting DRL for their applications [2], and we suggest a new set of them: 1) Establishing the application-side contribution (where the integration with the existing system is highlighted), 2) Defining the impact of problem

[8] E.g. Bayesian Deep Q-Networks: https://github.com/kazizzad/BDQN-MxNet-Gluon

framing (observation space design and task decomposition), 3) Advanced model understanding (spanning model configuration choices and interpretability) and 4) Establishing a solid training process (for which organized learning requires study). Among the research goals stemming from our revised challenges, we seek to understand some of the following aspects: the role of Bayesian RL for providing uncertainty estimates, such that a standard solution is chosen when the uncertainty is too high; improvements in the life-cycle design; work in facilitating the study of hierarchical task decompositions; and evaluations on the role of organized learning.

Moving forward, based on our progress and continued research requirements, we will pursue an extension to our project for Future SOC Lab resource usage.

Table 1: Estimated current completion of project goals

Goal	% done
Framework integration tool	100
Frameworks comparison	100
Overall partitioning solution	70
Offline-online life-cycle	60
Large action spaces	40
Safety unit tests	70
Model understanding tool	20

References

[1] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare. "Dopamine: A research framework for deep reinforcement learning". In: *CoRR* (2018). arXiv: 1812.06110 [cs.LG].

[2] G. C. Durand. *Production-Ready Learning-Augmented Data Management with Deep Reinforcement Learning*. Small Entry Accepted for ECML PKDD Workshop, Unpublished. Sept. 2019. URL: https://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/campero2019production.pdf (last accessed 2020-01-01).

[3] G. C. Durand, R. Piriyev, M. Pinnecke, D. Broneske, B. Gurumurthy, and G. Saake. "Automated Vertical Partitioning with Deep Reinforcement Learning". In: *European Conference on Advances in Databases and Information Systems*. Springer. 2019, pages 126–134.

[4] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, et al. "An introduction to deep reinforcement learning". In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pages 219–354.

[5] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, X. Ye, Z. Chen, and S. Fujimoto. "Horizon: Facebook's Open Source Applied Reinforcement Learning Platform". In: *CoRR* (2018). arXiv: 1811.00260.

[6] A. Haj-Ali, N. K. Ahmed, T. Willke, J. Gonzalez, K. Asanovic, and I. Stoica. *A View on Deep Reinforcement Learning in System Optimization*. 2019. arXiv: 1908.01275 [cs.LG].

[7] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. "Learned cardinalities: Estimating correlated joins with deep learning". In: *CoRR* (2018). arXiv: 1809.00677.

[8] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg. "Ai safety gridworlds". In: *CoRR* (2017). arXiv: 1711.09883.

[9] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica. "Ray rllib: A composable and scalable reinforcement learning library". In: *CoRR* (2017). arXiv: 1712.09381.

[10] H. Mao, P. Negi, A. Narayan, H. Wang, J. Yang, H. Wang, R. Marcus, R. Addanki, M. Khani, S. He, V. Nathan, F. Cangialosi, S. B. Venkatakrishnan, W.-H. Weng, S. Han, T. Kraska, and M. Alizadeh. "Park: An Open Platform for Learning-Augmented Computer Systems". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing atari with deep reinforcement learning". In: *CoRR* (2013). arXiv: 1312.5602.

[12] C. Molnar. *Interpretable machine learning*. self-published, 2019. ISBN: 978-0-244-76852-2. URL: https://christophm.github.io/interpretable-ml-book/ (last accessed 2020-01-01).

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), page 484. DOI: 10.1038/nature16961.

[14] I. Stoica. "Solving System Problems with Machine Learning". In: *Studies in Informatics and Control* 28.2 (2019), pages 119–132. DOI: 10.24846/v28i2y201901.

Impact of TLB Shootdown Latency

Robert Kuban, Randolf Rotta, and Jörg Nolte

Distributed Systems & Operating Systems
Brandenburgische Technische Universität Cottbus-Senftenberg
{firstname.lastname}@b-tu.de

Current multicore architectures use a non-consistent TLB to cache mappings from virtual to physical addresses. In order to provide a consistent virtual address space for applications, the operating system must invalidate TLBs of remote CPUs when modifying the virtual address space. Typically, the initator of such an invalidation propagates the event by sequentially sending an interrupt to each receiver. On large multicore systems, the latency of the TLB shootdown is dominated by the linearly growing multicast latency. The project was aimed to evaluate the impact of low latency TLB shootdowns on different applications. In this report, first a mechanism for tree-based event propagation is discussed. Then an overview is given on why no positive results can be expected with the benchmarks we intended to use for evaluation.

1 Introduction

Most multicore architectures have caches for address translation entries, named translation lookaside buffer (TLB), but do not keep these caches consistent by hardware. Therefore, the operating system must invalidate stale entries whenever it modifies the virtual address space of an application. Typically, this is done using a TLB shootdowns [3, 8], which notifies every CPU used by the application to invalidate its TLB. In many implementation, for example in the Linux kernel, the initiator of a TLB shootdown sequentially notifies each receiver. This leads to a linear scaling in the number of receivers, resulting in TLB shootdown latencies in the ballpark of $40\mu s$ for large receiver groups.

As demonstrated in the context of Barrelfish [1], logarithmic scaling in the number of receivers can be archived by using tree-based mechanism for propagating the notification. Many approaches for tree-based propagation in shared memory use a prebuild tree topologies [1, 5]. However, maintaining an optimal tree topology can be expensive when the group of potential recievers change frequently. The approach discussed in the next section (2) avoids this by using an implicit tree topology based on an algorithm of Bruck et al. [4].

The remainder of the report is structured as following: Section 3 describes three types of applications that where considered for evaluation and the reasons why we refrained from carrying out the evaluation using these benchmarks. The final section (4) provides a short summary.

2 Low Latency TLB shootdown

In a previous project at the HPI Future SOC Lab, we showed that the latency of the *membarrier* system call can be decreased with the tree-based propagation mechanism based on an algorithm of Bruck et al. [4]. Therefore, we extended the Linux kernel by providing an alternative implementation of the function `smp_call_function_many`.

Algorithm Starting with the set of receivers S and a root node s, the set S is partitioned with $|S'| = \min(\lfloor 0.5n \rfloor, n-1)$ and $S'' = S \setminus S'$. After selecting a leader $s'' \in S''$, the notification is propagated recursively to S' and S'' by s and s'', respectively.

Historically, the *mprotect* system call has been used as a substitute for the expedited *membarrier* system call: When *mprotect* removes the write access to a page in the virtual address space of a process, it becomes necessary to invalidate this page's TLB entry on every core that executes a thread of the process, leading to an TLB shootdown. On the x86 architecture, the TLB shootdown handler also have memory barrier semantic, enabeling the use of *mprotect* as an improvised *membarrier* system call. Consequently, the latency of the *mprotect* system call has been evaluated in the scope of the previous project.

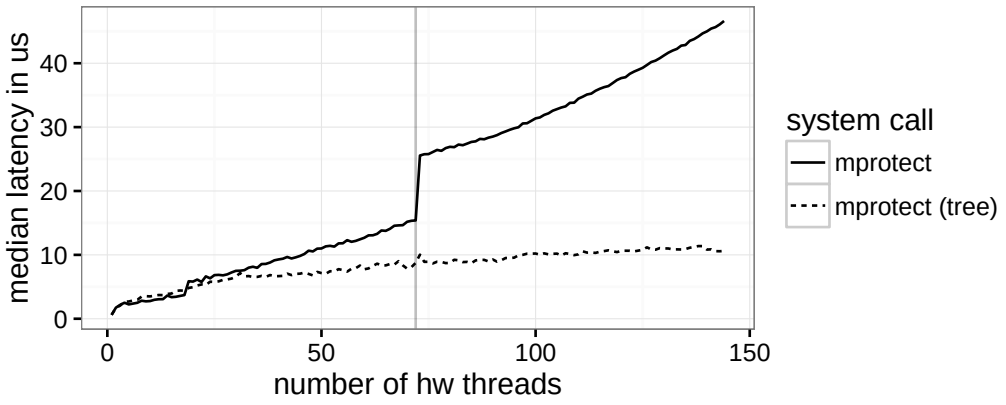


Figure 1: Latency of a TLB shootdown using using an sequential and a tree-based multicast. The horizontal line denotes 72 cores.

Figure 1 shows the latency of the *mprotect* system call for two variants: the sequential Linux implementation and our tree-based mechanism. Threads are pinned to cores, up to 72 threads only one hyperthread per core is used. The Linux implementation of the TLB shootdown scales roughly linearly with the number of cores. The tree-based implementation archives roughly logarithmic scaling, decreasing the latency for higher numbers of cores. This is achieved distributing the propagation overhead across multiple cores. However, this mechanism can not reduce the amount of work nessecary to propagate the TLB invalidation.

3 Applications

For the project, three types of applications where considered: **1**) Memory-intensive applications trigger TLB shootdowns on memory pressure by evicting pages from the page cache. **2**) Multithreaded MapReduce applications are interesting because there exist benchmarks that rely on page-level copy-on-write, which triggers an TLB shootdown on the first modification of a page. **3**) Various benchmark can be used to demonstrate that the propagation mechanisms does not degrade the performance of typical multithreaded applications.

3.1 Memory-intensive Benchmarks

Memory-intensive kernels, especially the one with low locality, can generate memory pressure which leads to evictions from the page cache. Before evicting a page, its mappings must be invalidated in all TLBs used by the application. The Parallel Research Kernels [9] are a collection of HPC compute kernels. We assumed that especially the sparse-matrix vector multiplication and the random access benchmark are sensitive to the latency of TLB shootdowns when under memory pressure.

Initial exploration have been made on a local 32-core machine. The amount of memory has been limited and a swap partition has been created in the remaining memory to simulate a system where DRAM is used as a cache for a slightly slower memory technology (NVRAM, SSD).

The benchmark used for our experiments consists of parallel writes to random pages. Unfortunately, even in this extreme scenario, the result suggests that mature caching algorithms, such as used to manage the Linux page cache, are well adapted against the high TLB shotdown latencies. Pooling of empty pages and prefetching can move the TLB shootdown out of the fast path. Consequently, memory-intensive application can not directly benefit from lower TLB shotdown latencies.

3.2 Multithreaded MapReduce

MapReduce applications, such as Phoenix [7] or Metis [6], typical map large files into memory. Similar to other memory-intensive application this may triggers page cache evictions, leading to TLB shootdowns carried out by the swapping daemon.

A special role has the *wordcount* benchmark application included in the Phoenix framework. As it maps the input file as copy-on-write and in-place converts any word to lowercase, it triggers a TLB invalidation for every page in the input file. Naturally, this is attractive for evaluating TLB consistency mechanisms. It is not completely clear if using copy-on-write for this benchmark was a conscious choice or is only an implementation artifact. Apart from this benchmark, copy-on-write does not appear to be widely used in applications.

For the parallel propagation mechanism, there is the additional problem that these frameworks use load balancing in the map phase. Parallelizing the propagation does not actually reduce the amount of work needed for propagation. As the productive

work is balanced between cores, redistributing the propagation work to speed up a single core is futile.

3.3 Other Parallel Benchmarks

In contrast to the applications discussed above, most parallel application avoid interaction with the virtual memory system. This is especially true for applications from the high performance computing (HPC) domain, for example application included in the Splash 2 [10] benchmark suite. However, even in Parsec [2], which specifically targets parallel non-HPC workloads, only one benchmark (dedup) is vm-intensive. Initial exploration unfortunately have shown no improvement in performance of dedup by solely decreasing the TLB shootdown latency.

As most of these applications interact as less as possible with the virtual memory subsystem, they can only be used to predict whether a TLB shootdown mechanism degrades the general system performance. Consequently, benchmarking these applications does not promise relevant results if no improvement can be shown for vm-intensive applications.

4 Summary

This project set out to evaluate the impact of a low latency TLB shootdowns mechanism on application performance. A tree-based propagation mechanism decreases the latency of TLB shootdowns by parallelizing the propagation, but does not decrease the amount of work necessary to invalidate the TLB. Further investigation into application benchmarks ruled out promising candidates for reasonable application benchmarks. Therefore, no resources of the HPI Future SOC Lab have been utilized.

Acknowledgement This work was supported by the German Research Foundation (DFG) under grant no. NO 625/7-2.

References

[1] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. "The Multikernel: A New OS Architecture for Scalable Multicore Systems". In: *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*. SOSP '09. Big Sky, Montana, USA: ACM, 2009, pages 29–44. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629579.

[2] C. Bienia, S. Kumar, J. P. Singh, and K. Li. "The PARSEC benchmark suite". In: *Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08*. New York, New York, USA: Association for Computing Machinery (ACM), 2008, pages 72–81. ISBN: 978-1-60558-282-5. DOI: 10.1145/1454115.1454128.

[3] D. L. Black, R. F. Rashid, D. B. Golub, and C. R. Hill. "Translation Lookaside Buffer Consistency: A Software Approach". In: *SIGARCH Comput. Archit. News* 17.2 (Apr. 1989), pages 113–122. ISSN: 0163-5964. DOI: 10.1145/68182.68193.

[4] J. Bruck, L. D. Coster, N. Dewulf, C.-T. Ho, and R. Lauwereins. "On the design and implementation of broadcast and global combine operations using the postal model". In: *IEEE Transactions on Parallel and Distributed Systems* 7.3 (Mar. 1996), pages 256–265. ISSN: 1045-9219. DOI: 10.1109/71.491579.

[5] S. Kaestle, R. Achermann, R. Haecki, M. Hoffmann, S. Ramos, and T. Roscoe. "Machine-Aware Atomic Broadcast Trees for Multicores". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pages 33–48. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/kaestle (last accessed 2020-01-01).

[6] Y. Mao, R. Morris, and F. Kaashoek. *Optimizing MapReduce for Multicore Architectures*. Technical report MIT-CSAIL-TR-2010-020. Messachusetts Intitute of Technology, 2010. HDL: 1721.1/54692.

[7] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. "Evaluating MapReduce for Multi-core and Multiprocessor Systems". In: *2007 IEEE 13th International Symposium on High Performance Computer Architecture*. Feb. 2007, pages 13–24. DOI: 10.1109/HPCA.2007.346181.

[8] P. J. Teller. "Translation-lookaside buffer consistency". In: *Computer* 23.6 (June 1990), pages 26–36. ISSN: 0018-9162. DOI: 10.1109/2.55498.

[9] R. V. der Wijngaart, T. Mattson, J. Hammond, S. Sridharan, and E. Georganas. *Parallel Research Kernels (PRK) v1.3*. Technical report. 2016. URL: https://github.com/ParRes/Kernels/raw/9e5d56c2fef0c5410a4d6237030e7ccfbe803efa/doc/parres-kern-report-v1.3.pdf (last accessed 2020-01-01).

[10] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. "The SPLASH-2 programs". In: *Proceedings of the 22nd annual international symposium on Computer architecture - ISCA '95*. June. New York, New York, USA: Association for Computing Machinery (ACM), 1995, pages 24–36. ISBN: 0-89791-698-0. DOI: 10.1145/223982.223990.

Current Technical Reports from the Hasso-Plattner-Institut

Vol.	ISBN	Title	Authors/Editors
157	978-3-86956-561-3	**Digital sovereignty: insights from Germany's education sector**	Christoph Meinel, Michael Galbas, David Hagebölling
156	978-3-86956-560-6	**Digitale Souveränität: Erkenntnisse aus dem deutschen Bildungssektor**	Christoph Meinel, Michael Galbas, David Hagebölling
155	978-3-86956-556-9	**Triple graph grammars for multi-version models**	Matthias Barkowsky, Holger Giese
154	978-3-86956-555-2	**Modular and incremental global model management with extended generalized discrimination networks**	Matthias Barkowsky, Holger Giese
153	978-3-86956-551-4	**Human pose estimation for decubitus prophylaxis**	Benedikt Weber
152	978-3-86956-550-7	**RailChain : Abschlussbericht**	Ingo Schwarzer, Said Weiß-Saoumi, Roland Kittel, Tobias Friedrich, Koraltan Kaynak, Cemil Durak, Andreas Isbarn, Jörg Diestel, Jens Knittel, Marquart Franz, Carlos Morra, Susanne Stahnke, Jens Braband, Johannes Dittmann, Stephan Griebel, Andreas Krampf, Martin Link, Matthias Müller, Jens Radestock, Leo Strub, Kai Bleeke, Leander Jehl, Rüdiger Kapitza, Ines Messadi, Stefan Schmidt, Signe Schwarz-Rüsch, Lukas Pirl, Robert Schmid, Dirk Friedenberger, Jossekin Beilharz, Arne Boockmeyer, Andreas Polze, Ralf Röhrig, Hendrik Schäbe, Ricky Thiermann
151	978-3-86956-547-7	**HPI Future SOC Lab – Proceedings 2018**	Christoph Meinel, Andreas Polze, Karsten Beins, Rolf Strotmann, Ulrich Seibold, Kurt Rödszus, Jürgen Müller (Hrsg.)
150	978-3-86956-546-0	**openHPI : 10 Jahre MOOCs am Hasso-Plattner-Institut**	Christoph Meinel, Christian Willems, Thomas Staubitz, Dominic Sauer, Christiane Hagedorn

ISBN 978-3-86956-564-4
ISSN 1613-5652